

AD-A119 252

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
ALR-46 COMPUTER GRAPHICS SYSTEM FOR THE ROBINS AFB ELECTRONIC W--ETC(U)
DEC 81 J W THAMES
AFIT/6CS/EE/81D-18

UNCLASSIFIED

NL

for 3

AD-A119 252

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

1992

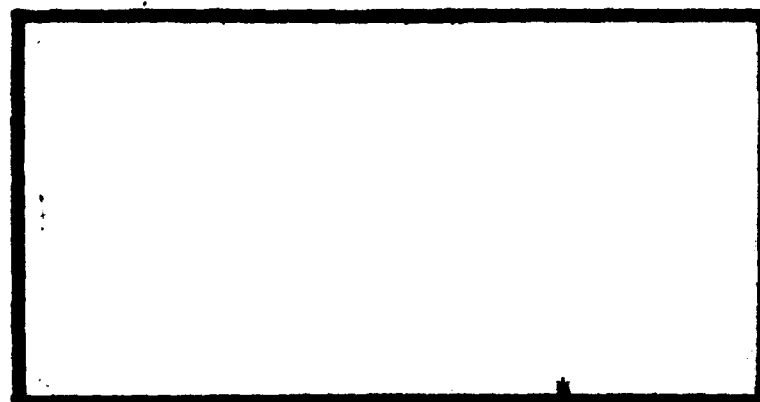
1992

1992

1992

AD A119252

DTIC FILE COPY



DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

82 09 18 088

DTIC
ELECTE
SEP 15 1982

H

RECEIVED
SEP 15 1982
AIR FORCE INSTITUTE OF TECHNOLOGY
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

AFIT/GCS/EE/81D-18

ALR-46 COMPUTER GRAPHICS SYSTEM
FOR THE ROBINS AFB
ELECTRONIC WARFARE DIVISION
ENGINEERING BRANCH LABORATORY

THESIS

AFIT/GCS/EE/81D-18 J. Wayne Thames
CIV USAF

DTIC
ELECTE
SEP 15 1982
H

Approved for public release; distribution unlimited.

AFIT/GCS/EE/81D-18

THE ALR-46 COMPUTER GRAPHICS SYSTEM
FOR THE ROBINS AFB
ELECTRONIC WARFARE DIVISION
ENGINEERING BRANCH LABORATORY

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

by

J. W. THAMES, BSEE

CIV USAF

Graduate Electrical Engineering

December 1981

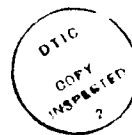
Approved for public release; distribution unlimited.

Preface

An ALR-46 computer graphics system was designed and implemented for the Electronic Warfare Division Engineering Branch Laboratory to reduce the time required to implement a change to the ALR-46 operational flight program.

I would like to express my deep appreciation to Dr. Gary B. Lamont, who as research advisor gave me valuable guidance and encouragement. Also, I thank my thesis readers, Dr. August Golden and Professor Charles Richard, whose constructive comments helped improve the clarity of this thesis.

Finally, I would like to thank Kirk for her help and encouragement during the past nine months.



Accession For	
NTIS GPA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

Contents

	Page
Preface	ii
List of Figures	v
List of Tables	vii
List of Acronyms and Abbreviations	viii
Abstract	ix
I. Introduction	1
Background	1
Environment	4
Problem	7
Scope	8
Summary of Current Knowledge	9
Approach	9
Thesis Development	11
II. System Requirements	13
Introduction	13
User Survey	13
System Level Requirements	13
Constraints on the ALR-46 CGS	14
Meeting the Requirements	16
System Component Architecture	21
Summary	42
III. Software Requirements	44
Introduction	44
Software Overview	44
Operator Interface	45
Parameter Capability	50
RWR Capability	52
Parameter / RWR Data	55
Selection of a Programming Language	58
Summary	59
IV. Design of CGS	61
Introduction	61
Logical Model	61
Physical Model	73

Contents (cont.)

Summary	78
V. Implementation and Testing	79
Introduction	79
Implementation	79
Testing	90
Summary	94
VI. Conclusions and Recommendations	95
Recommendations	96
Bibliography	98
Appendix A: ALR-46 Track File Word Format	100
Appendix B: ALR-46 Symbol / Type File	110
Appendix C: CGS Range and Azimuth Calculation	112
Appendix D: Module Structure Charts	114
Appendix E: Warnier-Orr Diagrams and CGS Source Code Listings . . .	121
Appendix F: CGS Data Dictionary	188
Appendix G: CGS Testing Documentation	210
Vita	231

List of Figures

Figure	Page
1 Current ALR-46 Support Station	5
2 Advanced ALR-46 Support Station	5
3 Advanced ALR-46 Support Station Network	6
4 System Requirements Verses Available Hardware	18
5 Pilot's ALR-46 System Display	25
6 Data Captured With the Hardware Monitor	32
7 Events Captured With the Hardware Monitor	34
8 PDP-11 Processor Compatibility	36
9 Unibus Configuration	37
10 Sponsor's PDP 11/34 Configuration	40
11 Sponsor's PDP 11/70 Configuration	41
12 AFIT's VAX-11/780 Configuration	42
13 Man / ALR46-CGS Interface	45
14 Parameter Mode User Display	51
15 Static ALR-46 System Display	53
16 RWR Mode User Display	55
17 ALR-46 Emitter Track File Link Structure	57
18 DFD Components	62
19 Context Diagram	64
20 System Diagram	65
21 Display Critical Track File Parameters (2.0) DFD.	67
22 Update CGS's Track File Entry (2.2) DFD	68
23 Simulate Pilot's Display (3.0) DFD	69

List of Figures (cont.)

Figure	Page
24 Extract / Format RWR Data (3.3) DFD	71
25 Help User (4.0) DFD	72
26 Factoring the Afferent Section	74
27 Factoring the Transform Section	75
28 Factoring the Efferent Section	75
29 Simulate Pilot's Display Module Structure Chart	76
30 Transaction Analysis Technique	76
31 Help User Module Structure Chart	77
32 Top-Down Implementation	80
33 Bottom_Up Implementation	82
34 Warnier-Orr Process Structure Diagram	85
35 Warnier-Orr System Overview	88
36 Warnier-Orr Diagram of the Parameter Process	91
37 Warnier-Orr Diagram of the RWR Process	92
38 Record Structure of the ALR-46 Emitter Track File	101
39 Symbol/Type File Structure	111
40 ALR-46 Coordinate System	113
41 System Structure Chart	115
42 Display Critical Track File Parameters Structure Chart	116
43 Update CGS's Track File Entry Structure Chart	117
44 Extract / Format Critical Parameters Structure Chart	119

List of Tables

Table		Page
1	List of Entries in the Emitter Track File	15
2	Address Limits Table	30
3	Event Address Table	33
4	Hardware Monitor Internal Storage Buffer	35
5	List of all DFD's	63
6	List of all Warnier-Orr Diagrams	86
7	Valid Keywords	89
8	Parameter Display Test Results	211
9	RWR Display Test Results	212

Abstract

An ALR-46 Computer Graphics System (CGS) for the Electronic Warfare Division Engineering Branch Laboratory was designed and implemented. The system is composed of the ALR-46 Radar Warning Receiver; real time hardware monitor; PDP 11/34 computer; and two display devices, a Tektronix model 4027 CRT and a Digital Equipment Corporation (DEC) model VT-100 CRT. The ALR-46 test engineers at the Engineering Branch Laboratory were interviewed; their functional requirements were translated into a detailed set of hardware and software requirements. Structured Analysis Techniques were used to produce a structured specification for the system requirements. Yourdon and Constantine's Transform and Transaction Analysis techniques were used to develop module structure charts for the software design. The final software design phase translated the module structure charts into Warnier-Orr (W/O) diagrams. These were translated into operational software using DEC Pascal. The software was implemented and tested using a top down approach. The modules and data structures were designed to allow additional capabilities to be added to CGS with minimum impact on the current system. The data acquired by the hardware monitor from the ALR-46 is passed to the PDP 11/34 for CGS analysis. This information is used either to simulate the pilot's display or to generate a critical parameter display describing the threats being processed by the ALR-46.

List of Acronyms and Abbreviations

ACM	American Computing Machine
AIAA	Advanced Integrated Support Station
AFLC	Air Force Logistics Command
ASD	Avionics Systems Division
ATS	Advanced Threat Simulator
A/D	Analog to Digital Converter
CGS	Computer Graphics System
CIR	Circle Command
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DEC	Digital Equipment Corporation
DECUS	Digital Equipment Corporation User Society
DFD	Data Flow Diagram
DMA	Direct Memory Address
DTIC	Defense Technical Information Center
ERA G	Erase Graphics Region Command
EW	Electronic Warfare
EWASIF	Electronic Warfare Avionics Integration Support Facility
EWOLS	Electronic Warfare Open Loop Simulator
GRA	Graphics Command
HIPO	Hierarchical-Input-Process-Output
HM	Hardware Monitor
IEE	Industrial Electronic Engineers
ISS	Integrated Support Station

List of Acronyms and Abbreviations (cont.)

K	One thousand and twenty four bytes
ML	Missile Launch
MTBF	Mean Time Between Failures
OFF	Operation Flight Program
PC	Program Counter
POL	Polygon Command
PRI	Pulse Repetition Interval
PW	Pulse Width
RF	Radio Frequency
RVE	Relative Vector Command
RWR	Radar Warning Receiver
SADT	Structured Analysis and Design Technique
SAM	Surface to Air Missile
STR	String Command
TAC	Tactical Air Command
VEC	Vector Command
WOR	Workspace Command
WR-ALC	Warner Robins Air Logistics Center
W/O	Warnier-Orr

I. Introduction

The Air Force supports many Electronic Warfare (EW) systems. The primary function of these systems is to respond to enemy action or potential enemy action (Ref 1:1-2). The support responsibility for each EW system is assigned to an Air Force organization based on the type of maintenance support. When a user of an EW system requests that a change be made, it is the responsibility of the supporting organization to implement the change and distribute the modified system. The Electronic Warfare Avionics Integration Support Facility (EWAISF) at Robins AFB is responsible for two classes of very sophisticated systems, Radar Warning Receivers (RWR) and Jammers. The time required to implement a change to either class of system is critical to the user and the EWAISF. The intent of this effort was to develop and implement a computer graphics system which would reduce the time required to evaluate and implement a system change.

Background

The primary function of the RWR is to warn the pilot of an enemy threat (anything that jeopardizes the life of the pilot or the successful completion of the mission). The pilot must take whatever action is necessary. The primary function of the jammer is to protect the pilot and insure the success of the mission by denying critical aircraft information such as aircraft heading, speed, position, etc., to the threat.

The early EW systems were built using all hardware components (Ref 1:2-6). The time required to make a change to the system was excessive. As a result, computers became an integral part of EW systems. A change to the system could be made by changing a few computer statements. Unfortunately, the personnel required to maintain the new, computer controlled EW systems required a knowledge of hardware, software, and electronic warfare. The Electronic Warfare Division at Robins AFB, Georgia was formed to provide the Air Force a centralized EW support facility.

The personnel assigned to the EWAISF provide both hardware and software support for Air Force Avionics Electronic Warfare systems. The ALR-46 system is the most widely used of all the EW system supported by the EWAISF. Two of the larger users of the system are the Strategic Air Command (SAC) and the Tactical Air Command (TAC). The ALR-46 was designed specifically to display the type and position of enemy threats. It is able to distinguish between different types of enemy threats by measuring specific parameters of the Radio Frequency (RF) energy received from the enemy threats and comparing the measured parameters to intelligence data parameters. If the parameters matched within a user defined tolerance, the appropriate threat symbol is displayed to the pilot. If the user of the system is not satisfied, he might request a change. Some of the reasons that can cause a user to request a change to the system are as follows:

- As the intelligence community increases its knowledge of enemy threats, new and better techniques for detecting the threats are

formed. The ALR-46 system has to be modified to install these new techniques.

- As is true with any computer controlled system, logical errors in the computer program do exist. When an error is found, the ALR-46 must be modified.
- Because of the vast number of enemy threats that exist, no one EW system is able to recognize every type of enemy threat. When a user of the ALR-46 system decides their system should recognize a new enemy threat, a change request is submitted to the EWAISF.

When a change request is received by the EWAISF (Ref 20), it is reviewed by the ALR-46 configuration control board and is entered, based on priority, into the list of modifications for the ALR-46. When a request for modification reaches the top of the queue, the request is assigned to an engineer. The engineer and assistants are responsible for making whatever changes to the system are required to satisfy the user's request. After the changes have been installed and tested, the updated system is forwarded to the user. The final phase of the system update was to summarize all of the documentation generated during the change process such as (1) who requested the change, (2) why was it needed, (3) what changes to the system were required to implement it, etc. The EWAISF maintains a history of the changes made to each EW system.

Environment

The ALR-46 section is maintaining their system on a 10-15 year old Integrated Support Station (ISS). Figure 1 depicts the current ISS that is currently used during the installation and testing of an ISS. Several steps are executed each time a modification is made to the ALR-46 system.

- The threat parameters are manually programmed into the threat simulator.
- The ALR-46 system is modified to meet the new user requirements. The user requirements could either cause a change in the flight program (software) or system hardware or both.
- Each modified portion of the system is tested followed by a test of the entire system. If the modification is extensive, the system will be flight-tested in the aircraft. Otherwise, the system will be bench-tested using either the local ISS threat simulator or the Electronic Warfare Open Loop Simulator (EWOLS) to simulate an enemy threat environment.

Since the flight processor is the only computer available to the ALR-46 test engineer, it has to be used to acquire and analyze data during the test phase of a modification. Data acquisition and analysis has a severe impact on the ALR-46 performance. The test engineer is limited to a few brief "snapshots" of the system. The failure rate of the current ISS, combined with its limited capability, forced the EWAISF

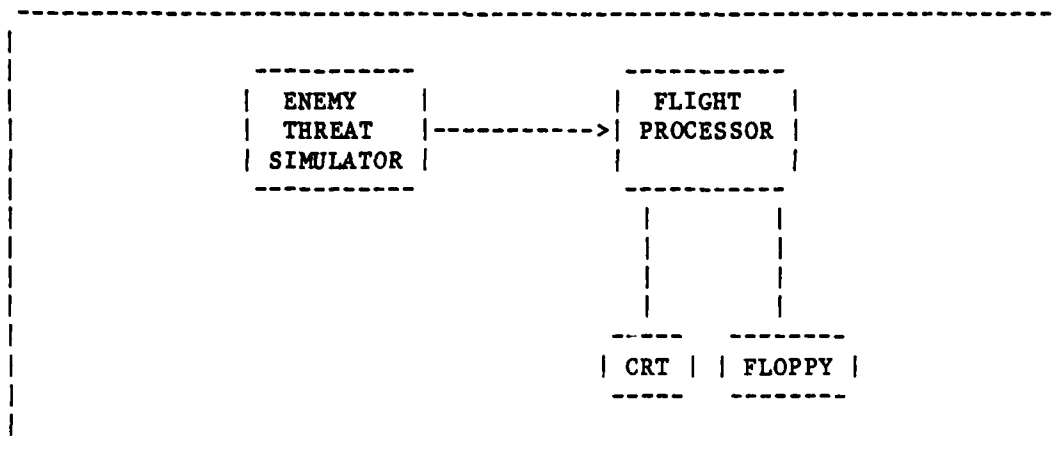


Figure 1. Current ALR-46 ISS

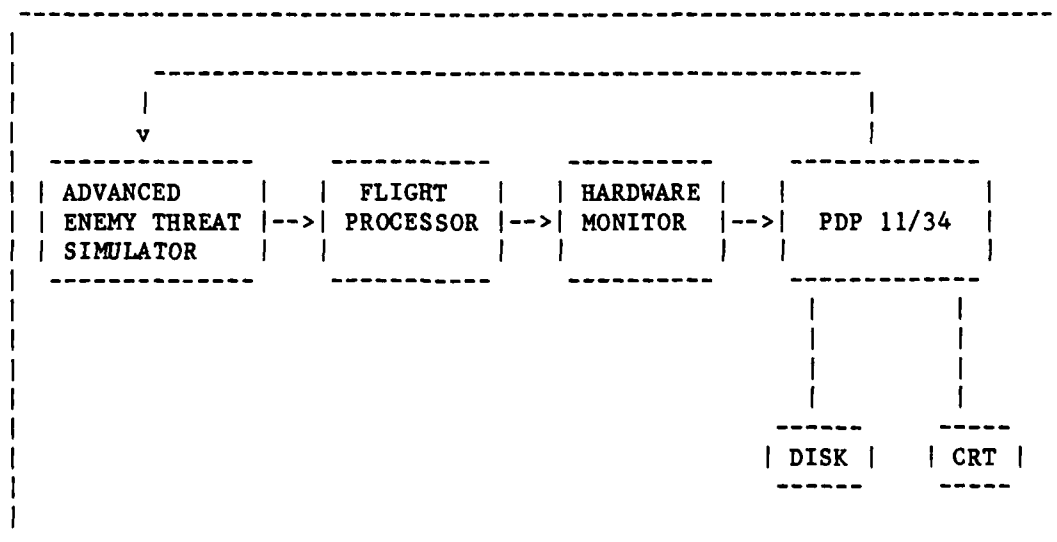


Figure 2. ALR-46 Advanced ISS (AISS)

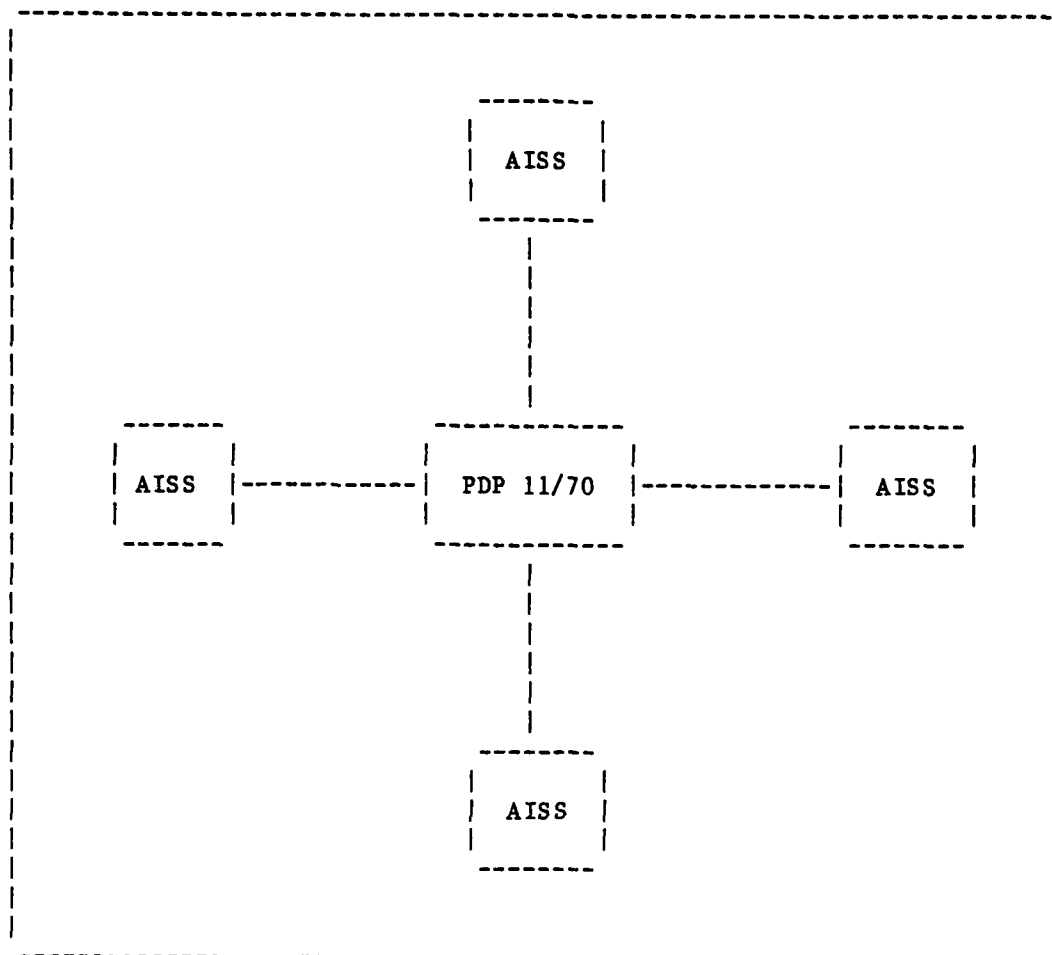


Fig 3. ALR-46 AISS Network

to replace the current ALR-46 ISS's with four advanced ISS's (Ref figure 2.). Each Advanced ISS (AISS) is equipped with a stand-alone PDP 11/34 computer that can be used to automatically program the advanced enemy threat simulator and, when used in conjunction with the hardware monitor, is capable of analyzing the performance of the ALR-46 system without affecting the performance of the flight processor. The four

AISS's will have access to a common data base located on the PDP 11/70 (Ref figure 3.). The PDP 11/70 will perform tasks either too large or too time-consuming for the PDP 11/34's. The first AISS and the PDP 11/70 was delivered to the EWASIF in July, 1981. Because of constraints beyond its control, the EWASIF was unable to develop/purchase all of the data acquisition and analysis programs it needed.

Problem

The lack of a means of analyzing data acquired by the AISS and reporting the results of the analysis to the user severely limits the AISS's capabilities. The only new capabilities available to the test engineer are as follows:

- a more advanced, computer controlled threat simulator
- a means of acquiring data from the flight processor in real time (hardware monitor)
- a more convenient means of editing and assembling the flight processor programs
- a proposed reduction in the mean time between failures (MTBF)

Even though the above capabilities will assist the test engineer when making a modification to the ALR-46 system, a further reduction in the time to test the system could be realized if the following existed: a closed loop test system in which a known set of threats would be simulated by the advanced threat simulator and the flight processor

would be monitored to determine which threats it recognized. The list of threats that were simulated would be compared to the threats recognized by the flight processor. The system could generate, for the test engineer, a list which contained (1) the threats that were simulated, (2) the threats that were recognized, and (3) the threats that should have been displayed. The magnitude of this test system is not apparent from this simple description. A mode of the ALR-46 is required to automatically produce a list of threats that should have been produced with a given combination of simulated threats. Modeling flaws have been the topic of many discussions at the EWAISF. The threat parameter tolerances that would be used to define whether the system had "accurately" recognized the threat is also a grey area among personnel at the EWAISF. For example, if the RF value programmed into the advanced threat simulator was 1000.6 and the ALR-46 system measured the signal at 985.3, is this right or wrong? Until the modeling and tolerance questions are resolved, this portion of the test system can not be completed.

Scope

This study will concentrate on the analysis, design, and implementation of a system to analyze and display the information captured by the hardware monitor. Since data can be accessed in any order by the OFP, the information provided by the hardware monitor may appear to be a series of binary words taken randomly from the ALR-46 central processing unit's (CPU) memory. The information generated by analyzing the data taken from the ALR-46's memory will be presented to

the operator in two forms: (1) a color reproduction of the pilot's display using a Tektronix 4027 CRT. The color will be utilized to distinguish between two or more threats of the same type, when being displayed simultaneously (2) a display of all of the information the ALR-46 system has acquired on each enemy threat. The completed system will be delivered to the sponsor (EWAISF) with a recommendation on whether or not the benefits of a color graphics CRT justify the additional expenditure on future AISS's.

Summary of Current Knowledge

The VAX-11/780, Tektronix 4027 CRT, and the DEC VT-100 are essential components in this research and the only knowledge of any of these systems comes from the vendor supplied manuals. For the VAX-11/780, this consist of a series of system manuals, the Pascal manuals (Refs 2 and 3), and DEC's VAX-11/780 book (Ref 4). The documentation on the Tektronix 4027 consist of an operator's manaul (Ref 5) and a programmers reference manual (Ref 6). The documentation on the VT-100 consist of a user's manual (Ref 7).

Approach

A complete system specification for the ALR-46 Computer Graphics System was developed. A "top down" and "structured" approach (Ref 8:279) was selected for the analysis and design of the graphics system. A top down strategy allowed the design to first address those levels closest to the user and thereby insure that the design was consistent with the user's requirements. Each lower level reveals more and more

details about the system. Data Flow Diagrams (DFD's) were used as an aid in the design phase of the system.

The first phase of the investigation consisted primary of researching the literature on similar systems to gain a better working knowledge. Exhaustive searches of American Computing Machine (ACM), Digital Equipment Corporation User Society (DECUS), Industrial Electronic Engineers (IEE), Defense Technical Information Center (DTIC), AFIT Theses, Avionics System Division (ASD), Avionics Labs, and the EWAISF failed to produce a similar system. The research did provide information on man / machine interfaces.

Next, six ALR-46 engineers were interviewed. From these interviews, a set of functional requirements were compiled and these were used to help derive the system requirements for the ALR-46 Computer Graphics System (CGS). The software requirements were documented using Structured Analysis (Ref. 8). The structured specifications consisted of a series of data flow diagrams. Yourdon and Constantine's structured design technique (Ref. 17) was used to translate the structured specification into module structure charts. The final phase of the CGS design was to generate Warnier-Orr (W/O) diagrams from the module structure charts. The W/O diagrams were used to develop the CGS software in DEC Pascal on a VAX-11/780. A top-down approach was used in the implementation and validation of CGS.

Thesis Development

The development of this thesis followed the approach that was taken in the investigation. The CGS functional requirements were analyzed in Chapter II. This chapter translated the results of the user survey into the system level requirements. Also included was introductory information on the ALR-46 system, hardware monitor, Tektronix 4027 CRT, and the DEC VT-100 CRT.

Chapter III further defined the CGS requirements. The operator interface, Parameter capability and RWR capability were each described in more detail along with the selection of the CGS language. Appendices A, B, and C provide supporting information including a complete description of the ALR-46 data which must be processed by CGS.

Chapter IV refined the CGS software requirements further using Structured Analysis (Ref 8). The data flow diagrams produced with Structured Analysis were translated into module structure charts using Yourdon and Constantine's structured design technique (Ref. 17). A complete set of module structure charts for CGS is contained in Appendix D.

Chapter V described the implementation and testing of the CGS. In this chapter, the factors affecting implementation were described, as well as the testing techniques employed. The Warnier-Orr diagrams and source code listings are in Appendix E while Appendix F contains the CGS data dictionary. The test data is contained in Appendix G.

Finally, Chapter VI summarized this investigation and gave recommendations for follow-on research efforts.

II. ALR-46 CGS Functional Requirements and System Configuration

Introduction

The first phase of any system development is to define requirements. The development of the CGS functional requirements is described in this chapter. The results of the user survey are described followed by a description of the requirements and constraints developed from the results of the survey. In the next section, the requirements and constraints are defined in terms of an actual computer system. Finally, each component in the computer system is discussed. A more detailed analysis of the CGS software requirements can be found in Chapter III.

User Survey

The requirements of the ALR-46 CGS had to be further defined to insure that the system produced would meet the primary goal of the EWAISF, which was to reduce the time required to modify the ALR-46 system. The following personnel assigned to the ALR-46 were interviewed: 4 Electronic engineers, 2 technicians, 1 unit chief, and 1 section chief. The survey was divided into three major sections.

In the first section, the users were asked what phase of modification to the ALR-46 was the most time consuming. All of the users overwhelmingly agreed that the test/debug phase was the most time consuming.

In the second section, the users were asked to describe from a hardware and operator interface point of view, how a reduction in test/debug time could best be achieved. The requirements identified by the users are described in the next section. The users were unable to prioritize the requirements because each requirement was considered essential for the particular type of testing it would be used with. However, all of the users did agree that the parameter display capability would be used the most often.

In the third section, the users were asked to describe their requirements for an operator interface. The requirements identified by the users are described in the next section.

System Level Requirements

The user survey results and the author's experience were used to define the following system requirements:

1. RWR Display. A means of reproducing the RWR display in real time (the information being displayed on the simulated RWR display can not lag the actual pilot's RWR display by more than two second). If two threats with identical symbols are displayed simultaneously, each symbol will be assigned a different color.

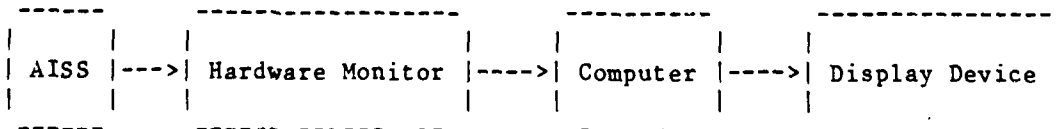
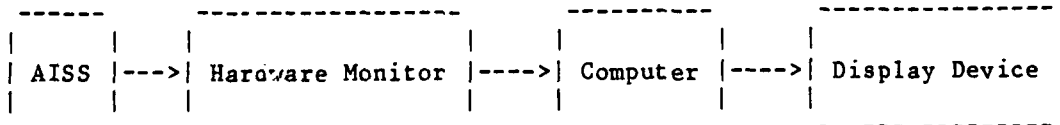


Table 1. Critical Emitter Track File Entries

ENTRY		NUMBER OF CHARACTERS
1.	SYMBOL	4
2.	TYPE	4
3.	BANDS	5
4.	PRI OR FRAME PERIOD	6
5.	PULSE TRAIN DESCRIPTOR	4
6.	PRI AVERAGE	5
7.	DEVIATION	5
8.	SPECIAL	2
9.	POWER	1
10.	POWER ROUND	1
11.	HIGH BAND SINE	3
12.	HIGH BAND COSINE	3
13.	CEPC	1
14.	AGE COUNT	2
15.	C/D	2
16.	PRIORITY	2
17.	PRIORITY POWER	1
18.	PRIORITY POWER ROUND	1
19.	AZIMUTH	3
20.	LETHALITY RING	3
21.	M.L. AGE	1
22.	RECORD NUMBER	2

2. Parameter Display A means of displaying critical threat file parameters (Ref Table 1).



3. Operator Interface. The operator interface must have the following capabilities: (a) It must have a "beginner" and an

"experienced" mode of operation. The "beginner" mode uses very descriptive prompts that lead the inexperienced operator through the prompting session. The "experienced" mode uses very concise prompts that help to keep the experienced user from becoming bored during the prompting session, (b) The interface must be very "forgiving". That is, the system should not abort if the user makes a mistake, it should resume at the appropriate level, (c) The terminology used in the prompts must be as similar as possible to that used by the user at the present time, and (d) A help command should exist at each major level of the prompt structure.

4. Operational Environment. Each of the four AISS's must be able to operate the CGS simultaneously.
5. Application Environment. The CGS must operate in the same environment as the ALR-46 system and must be readily available. Since space is very limited in the ALR-46 test environment, the space required by the CGS must be minimal.

Constraints on the ALR-46 CGS.

1. Compatibility. Another requirement for the ALR-46 CGS was that it must be compatible with DEC's PDP-11 series of computers, specifically the PDP 11/34 and the PDP 11/70. The EWAISF defined DEC computers as the standard for use on all support equipment.
2. Operational Environment. The CGS must not affect the operation of the ALR-46. If the ALR-46 is affected by the operation of the CGS,

the results of the test being performed would be biased.

3. Cost. Any hardware or software that already exist could be used in order to minimize the cost of the CGS. One of the two color graphics terminals (Chromatics model 5001 and the Tektronix model 4027), available from the sponsor, should be used if possible.
4. Programming Language. The software can be written in FORTRAN or Pascal. Both languages satisfy the standardization requirements of the sponsor. Assembly language will only be used if it is essential for execution speed.

Meeting the Requirements.

Introduction. In this section, the hardware requirements defined in the previous section are translated into a physically realizable system. The software requirements, including the operator interface, are discussed in Chapter III. The first step in translating requirements into a physical system was to determine what new equipment was needed by comparing the requirements and the available hardware (Ref figure 4). This made it apparent that a PDP 11/34 computer, a hardware monitor, and a DEC VT100 CRT, all of which are already in each AISS, combined with the Tektronix 4027 CRT, would satisfy all of the CGS hardware requirements. Even though the Chromatics 5001 is a color graphics terminal, it was eliminated from consideration because it does not support the requirements of the operator interface. It does not allow the screen to be divided into regions. The PDP 11/70 was

REQUIREMENTS	AVAILABLE HARDWARE				
	PDP-11/70	PDP-11/34	VT-100	Chromatics 5001	Tektronix 4027
RWR Display		X			X
Parameter Display		X	X		X
Operational Environment		X	X		X
Application Environment	X	X	X	X	X

CONSTRAINTS	AVAILABLE HARDWARE				
	PDP-11/70	PDP-11/34	VT-100	Chromatics 5001	Tektronix 4027
Compatibility	X	X			
Operational Environment	X	X	X	X	X
Programing Language	X	X	X		X

Figure 4. Systems Requirements Verses Available Hardware

eliminated as a potential host for CGS because it did not satisfy the real time constraints of the first two requirements (RWR Display and Parameter Display). The additional time required to move the data

acquired by the hardware monitor from the PDP 11/34 to the PDP 11/70 would cause an excessive time delay between the ALR-46 display and the CGS display. The requirements/constraints verses the CGS hardware configuration are discussed below:

1. RWR Display. The Tektronix 4027 is more than capable of reproducing the RWR display. With a list of 64 colors to choose from (Ref 5), the operator will not have a problem distinguishing between threat symbols. The 4027 was selected to simulate the pilot's display because of its color graphics capabilities (described in a later section) and because there were no expenditures or long lead times since the sponsor already owned it. The hardware monitor will be used to acquire the data from the ALR-46, and the PDP 11/34 will be used to host the software needed to analyze the data supplied by the hardware monitor and to drive the Tektronix 4027 CRT.
2. Parameter Display. The VT100 CRT will be used to display all of the critical parameters (Ref Table 1) the operator needs during the test/debug phases of an ALR-46 modification. The VT100 was selected as the parameter mode display device because of its cursor control capabilities (described in a later section) and because there were no expenditures or long lead times since it was already a part of the AISS. The hardware monitor will be used to acquire data from the ALR-46, and the PDP 11/34 will be used to host the software needed to analyze the data supplied by the hardware monitor and to drive the VT100 CRT.

3. Operator Interface. Since satisfying the requirements of the operator interface dealt mostly with the design of the CGS software package, they will be discussed in Chapter III.
4. Operational Environment. Using the PDP 11/34's and the VT100's to host the CGS, easily satisfies this requirement, except when two users want to simultaneously exercise the RWR display capability of CGS. There is no immediate solution to this problem since the sponsor only has one Tektronix 4027. The hardware monitor, discussed in a later section, provides CGS with a means of acquiring the data it needs without affecting the operation of the ALR-46.
5. Application Environment. Again, using the PDP 11/34's and the VT100's to host the CGS easily satisfy this requirement, since these devices are already an integral part of the AISS. The Tektronix 4027 will be easily transportable to the required AISS.
6. Compatibility. The PDP 11/34 and VT100's definitely meet the sponsors compatibility requirements. They are already an integral part of the ALR-46 AISS. The Tektronix 4027 CRT is also fully compatible with DEC 's PDP-11 and VAX family of computers (Ref 5).
7. Cost. No equipment, in addition to that already owned by the sponsor, was needed to build the CGS.
8. Programming Language. The PDP 11/34 supports both Pascal and FORTRAN. The selection of the language used to develop CGS is discussed in Chapter III.

System Component Architecture.

In the following section, the ALR-46, VT100 CRT, Tektronix 4027, hardware monitor, threat simulator, PDP 11/34, PDP 11/70, and VAX-11/780 will be described followed by a description of the features that all three computers have in common. This section will not only provide the reader with a clearer understanding of CGS, but will also emphasize the compatibility that exist between the three computers. This compatibility made it possible for CGS to be developed on the VAX-11/780 and then transfered to the PDP 11/34.

ALR-46 System. This section discussed the ALR-46 system and should provide a general idea of how the ALR-46 functions, as well as an idea of the magnitude of the problem the test engineer is faced with in validating the system after a modification has been made. But for a description of the ALR-46 system to be meaningful, the EW environment the ALR-46 operates in must be described.

The EW Environment consist of one or more enemy threats. The threats can be divided into classes or types. The ALR-46 uses the characteristics of the signals that each threat transmits to identify it. The threat characteristics, also called parameters, are defined below:

- A. Pulse -- used to gate the RF energy being transmitted by the threat.
- B. Pulse Width (PW) -- the length of time the RF energy is transmitted.

C. Pulse Repetition Interval (PRI) -- the time between pulses.

D. Scan Pattern -- the transmitter is moved back and forth to cover more area. This movement creates a specific pattern that can be identified by the ALR-46 system.

E. Radio Frequency (RF) -- each threat transmits at a given RF. The EW system uses this to aid in the identification of the threat.

F. Power Level -- the amount of RF energy measured at the ALR-46.

When the energy transmitted by the threat strikes the plane, a small portion of the energy is reflected back to the enemy threat where it is received. The enemy threat analyzes the energy reflected back from the plane (target) to determine the distance between the threat and the aircraft (range), the direction of the aircraft (azimuth), altitude, speed, etc. The ALR-46 system on the plane also receives the RF energy transmitted from the threat.

The front end of the ALR-46 consist of four receivers (ports), RF measurement equipment, and a series of analog to digital converters (A/D's). The four receivers perform two major functions. First and foremost, they received a portion of the RF energy transmitted by the enemy threat. Because they are located at different points on the plane (normally one on each wing tip, one on the front and one on the tail of the plane), they can be used by the ALR-46 system to define the direction of the threat (azimuth) relative to the heading of the aircraft.

A pulse of RF energy is received simultaneously at each of the four receivers. The time the pulse arrived is recorded, the power level of the signal at each receiver is also measured and recorded, and the frequency is measured and recorded. The time of arrival (TOA), frequency, and power levels are digitally transmitted to the ALR-46 flight processor.

The flight processor is the heart of the ALR-46 system. It controls the operation of the entire system. It is responsible for commanding the front end to make measurements, analyzing the parameter data returned from the front end, and displaying, to the pilot, the status of the EW environment. The software in the processor can be divided into three categories:

- (1) Threat identification tables -- The threat ID tables contain a list of parameter values for each threat the ALR-46 system is able to recognize.
- (2) Emitter Track File -- The emitter track file is a collection of data on each threat the ALR-46 system has identified. Of course, this list is constantly changing because new threats begin transmitting, current threats shut off, and more information is derived about each threat (PRI, scan, etc.). The list is sorted by threat priority (the highest priority threat points to the next higher priority threat, etc.).
- (3) The Parameter Algorithms -- These algorithms perform the actual processing done by the system. Separate algorithms are used to

calculate PRI, scan, etc. When the algorithm calculates a threat parameter, it is entered into the emitter track file.

When an emitter track file entry, such as range, azimuth, etc. is changed, the appropriate changes have to be made to the pilot's display. For example, as the plane flies closer to a threat, the symbol representing that threat is moved closer to the center of the display (Ref figure 5). The center of the display represents the aircraft's position.

Many interim test are performed while a modification to the ALR-46 is being made. The pilot's display is the only means the test engineer has of determining if the system is working correctly unless the flight processor is stopped and the emitter track file is dumped to a CRT. In 1981, ALR-46 engineers use a large amount of time analyzing a given enemy threat symbol being displayed on the ALR-46 monitor. Personnel at the EWALSF feel that after a modification has been made, the time required to test the ALR-46 system can be reduced through the use of a color display (Ref figure 5) or if critical entries of the emitter track file (Ref Table 1) were available in realtime (no more than two seconds delay between actual events and the event being displayed).

DEC VT100 CRT. The VT100 CRT serves as an input device for operator commands to the computer and as an output device for data sent from the host computer to the operator. Time consuming preventive maintenance test are not necessary because the VT100 has built in diagnostics. Each time the VT100 is turned on, the advanced video

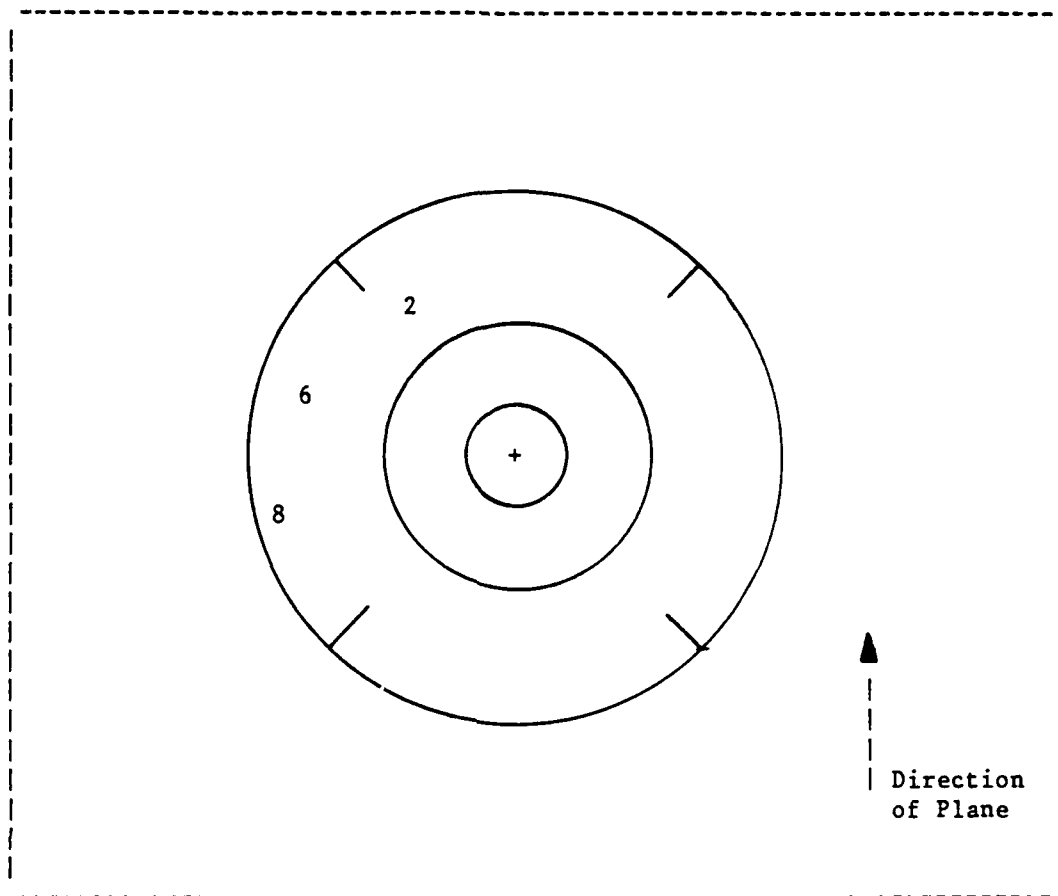


Figure 5. Pilot's ALR-46 System Display

Note: Each symbol displayed more than once is assigned a unique color.

memory, nonvolatile memory, internal memory, and keyboard are checked. Unlike most terminals, the VT100 does not use switches or jumpers to individually turn the built-in terminal features on or off (Ref 7:8). Instead, the VT100 uses a nonvolatile memory which always remembers what features have been selected. The selection and storage of built-in terminal features is performed by simply pressing the "set-up" key and selecting the desired features. Some of the features that can be

controlled are autorepeat, light or dark background, margin bell, keyclick, baud rate, tab stops, etc.

With the advanced video option, the VT100 can display up to 24 lines of 132 characters each. The VT100 can highlight any character(s) on the screen in any of the following ways: bold, blink, underline, reverse, or any combination of these (Ref 7:65).

The VT100 has a large selection of programmable features that can be controlled from the host computer. Some of the features applicable to CGS are described below:

1. Cursor Position Report (CPR) - The CPR command reports the active position of the cursor by returning its line and column number.
2. Cursor Backward (CUB) - The CUB command moves the sursor to the left the number of spaces requested.
3. Cursor Down (CUD) - The CUD command moves the cursor downward the number of lines specified without affecting the column position.
4. Cursor Forward (CUF) - The CUF command moves the cursor to the right the number of spaces requested.
5. Cursor Position (CUP) - The CUP command moves the cursor to the position specified independent of the current position of the cursor.
6. Cursor Up (CUU) - The CUU command moves the cursor up the number of lines specified without affecting the column position.

7. Double Height Line (DECDHL) - The DECDHL command allows the user to create lines of text that are twice as high as normal.
8. Double Width Line (DECDWL) - The DECDWL command allows the user to create lines of text that are twice as wide as normal.
9. Set Top and Bottom Margins (DECSTBM) - The DECSTBM command is used to set the top and bottom margins which define the scrolling region. Part of the screen can remain stable while the remainder of the screen can be used for operator input and output.
10. Character Attributes - This command is used to emphasize a character or a series of characters by turning on bold, underscore, blink reverse video, or any combination of these.

All of the VT100 features described make it an excellent choice for use in the ALR-46 CGS.

Tektronix 4027 Color Graphics CRT. The 4027 belongs to the class of machines popularly known as "smart terminals". Like the VT100, the 4027 is used to carry communications between the operator and the host computer. In addition, the 4027 contains its own microprocessor and supporting electronics (Ref 7:1-1). With these electronics, the 4027 responds to its own set of commands, independently of the host computer. The 4027 is not intended to be a stand-alone computing system. Rather, its computing ability complements that of the host computer, enabling the user to make full use of the 4027's information display capabilities. It is designed especially for creating color graphic

displays in up to 64 different colors. The 4027 consist of a display unit and a keyboard attached to the display unit by a thin cable. The display unit contains a 13 inch color CRT capable of displaying 34 lines of 80 column text. The 4027 display memory can be divided into two portions. One portion, called the workspace, serves as a composition area for creating color graphics, editing text files, filling out forms, or displaying the results of application programs (Ref 7:1-2). The monitor portion of memory stores messages to and from the computer. Each time the 4027 is turned on, all of its memory is checked by internal diagnostics. Many of the operating parameters, such as baud rate, left margin, right margin, tab stops, parity, etc., can be set from the keyboard or from the host computer. The 4027 has a large host of programmable features that can be controlled from the host computer. Some of the features applicable to CGS are described below:

1. Jump Command - This command positions the cursor to the row and column specified, independent of the cursor's current position.
2. Up Command - This command moves the cursor up the number of lines specified.
3. Right Command - This command moves the cursor to the right the number of columns specified.
4. Left Command - This command moves the cursor to the left the number of columns specified.
5. Color Command - This command is used to designate the color of

subsequent graphics.

6. Map Command - The Map command provides a selection of 64 possible colors of which eight may be designated at any one time.
7. Circle Command (CIR) - The CIR command is used to create circles, circle sectors, and equilateral polygons.
8. Graphic Command (GRA) - The GRA command is used to define a graphic region that will later be used to display graphics.
9. Vector Command (VEC) - The VEC command is used to draw line segments in the graphic region.
10. Relative Vector Command (RVE) - The RVE command is used to draw vectors by specifying relative coordinates. A line is drawn starting at the last graphic cursor position.
11. Polygon command (POL) - The POL command is used to draw a large number of colored shapes and panels.
12. String Command (STR) - The STR command is used to write text into the graphic region.
13. Erase G Command (ERA G) - The ERA G command is used to erase a graphic region.
14. Workspace Command (WOR) - The WOR command is used to define the size of the workspace. The number of lines not defined as workspace is used by the monitor.

Table 2. Address Limits Table

Lower Limit	1
Upper Limit	1
Lower Limit	2
Upper Limit	2
...	
Lower Limit	n
Upper Limit	n

All of the functions described are performed by the 4027's internal processor, thus enabling the host processor to simultaneously perform other task. The only problem encountered with the 4027 is that it failed three times in a four month period. Tektronix has assured the sponsor the problems with the 4027 have been resolved.

Hardware Monitor. The hardware monitor (HM) is an interface between the ALR-46 and the PDP 11/34. The monitor is used to extract

data from the ALR-46 in real time (within a few microseconds of when it actually occurred) and transfer it to the PDP 11/34 without affecting the operation of the ALR-46 (Ref 13:3-1). Data captured by the hardware monitor is stored in the HM's 2048 word internal storage buffer until the monitor can gain control of the PDP 11/34's unibus (unibus is described below). When access has been gained to the unibus, the data is transferred by direct memory access (DMA), i.e., the data in the HM can be transferred into the PDP 11/34's memory without the aid of the PDP 11/34's central processing unit (CPU). The DMA process is sometimes called "cycle stealing" because the HM steals cycles from the CPU (Ref 4:224). The hardware monitor has two modes of operation. In the first mode, a series of address limits (Ref Table 2), that are to be monitored, are defined by the user. Each time an address, that falls into one of the address limits, is written to, the address and its contents are saved in the monitor's storage buffer (Ref 13:4-40). As soon as there is time, the monitor transfers the data in the storage buffer to the PDP 11/34 (Ref figure 6). In the second mode, a series of event addresses (Ref Table 3), that are to be monitored, are defined by the user. The event addresses represents addresses of parts of the ALR-46 operational flight program (OFF). The program counter (PC) is a pointer to the specific part of the OFF being executed. Each time the contents of the program counter matches one of the entries in the event address table (Ref Table 3), the contents of the OFF's program counter and the contents of the real time clock (two words) are temporarily saved in the monitor storage buffer (Ref 13:4-40). The value "177777" is used

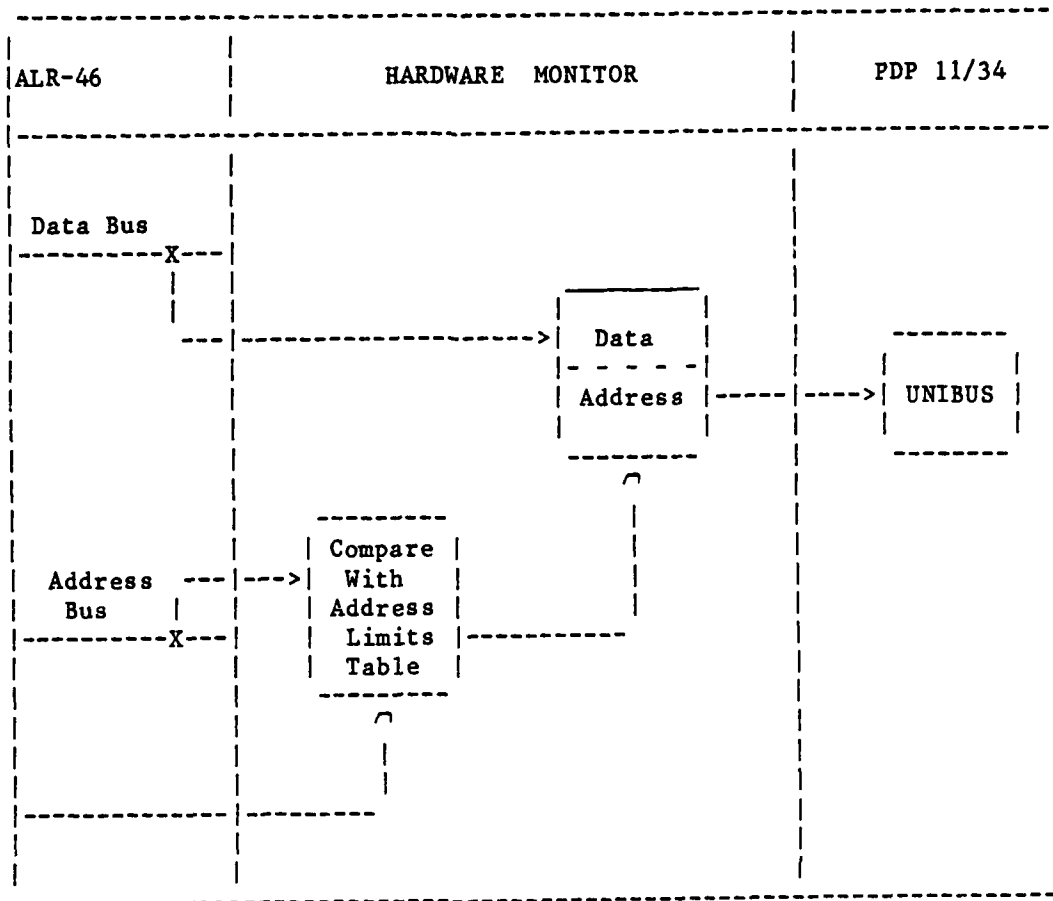


Figure 6. Data Capture With the Hardware Monitor

as a marker to distinguish between the PC and time (Ref fig 7). As soon as there is time, the monitor transfers the information in the storage buffer to the PDP 11/34 (Ref figure 7).

Both hardware monitor modes can be exercised simultaneously without affecting the operation of the ALR-46 in any way. The HM's internal storage buffer would look similar to table 4 if both modes were engaged at the same time.

Table 3 Event Address Table

The diagram illustrates a program counter stack. It consists of a vertical column of rectangular boxes, each representing a program counter entry. The entries are labeled as follows:

- Program Counter 1
- Program Counter 2
- Program Counter 3
- ...
- Program Counter n-1
- Program Counter n

Each entry is separated from the next by a dashed horizontal line. The stack is bounded by vertical lines on the left and right sides.

Threat Simulator. The AISS threat simulator is used to stimulate the ALR-46. It creates an EW environment where the ALR-46 can not tell the difference between the simulated EW environment and the actual EW environment.

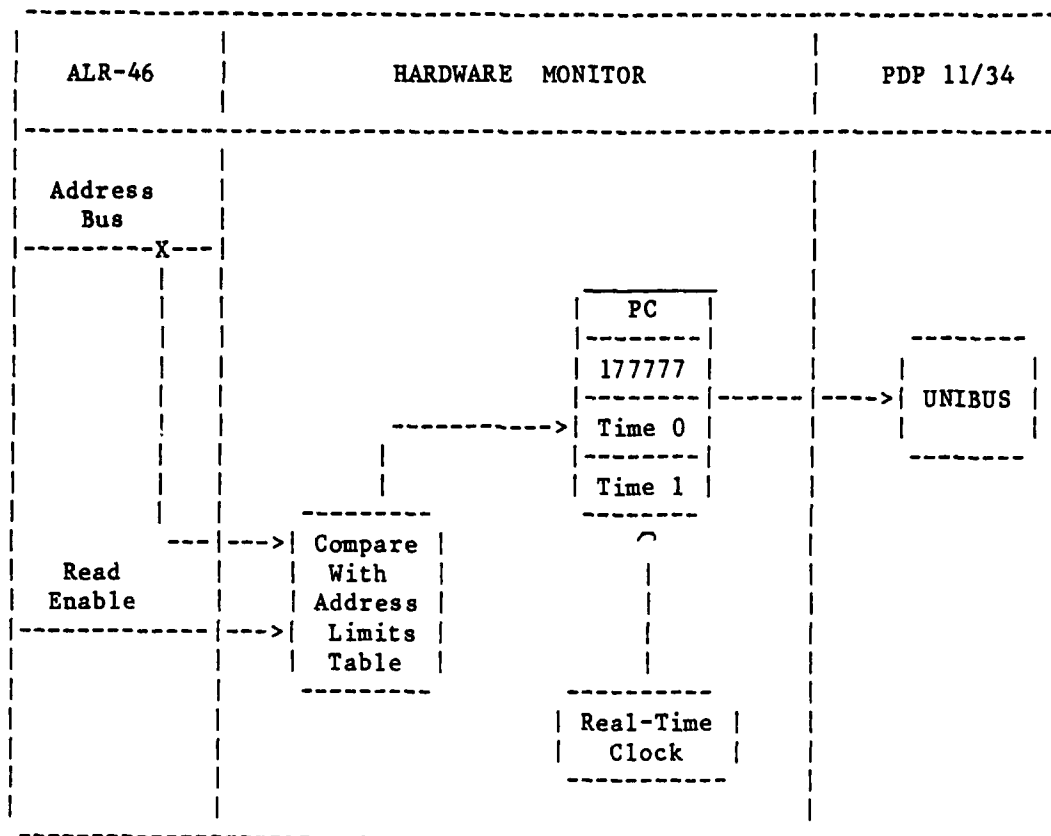


Figure 7. Event Capture With the Hardware Monitor

PDP-11 Computer. The PDP-11 processors are a family based on common architecture. Compatibility is inherent in design, and is reflected in the software and in the peripheral options (Ref 9:1). For example, the VT100 terminal is compatible with the PDP 11/34, PDP 11/70, and the VAX-11/780 processors. The RSX-11M operating system is also compatible with the PDP 11/34 and the PDP 11/70. The compatibility of the PDP 11 and VAX-11 processors can be seen in Figure 8.

Table 4. Hardware Monitor Internal Storage Buffer

```

Data
-----
Ram Address
-----
Data
-----
Ram Address
-----
PC
-----
177777
-----
Time 0
-----
Time 1
-----
~~~~~
~~~~~
~~~~~
~~~~~
~~~~~
~~~~~
      :
      :
      :
      :
      :
      :
      :
~~~~~
~~~~~
~~~~~
~~~~~
~~~~~
~~~~~
-----
Data
-----
Ram Address
-----
Data
-----
Ram Address

```

The PDP-11 processors have one outstanding characteristic in common: they all process data on a data bus called the "Unibus". The "Unibus" allows communication between any two devices on the bus. It

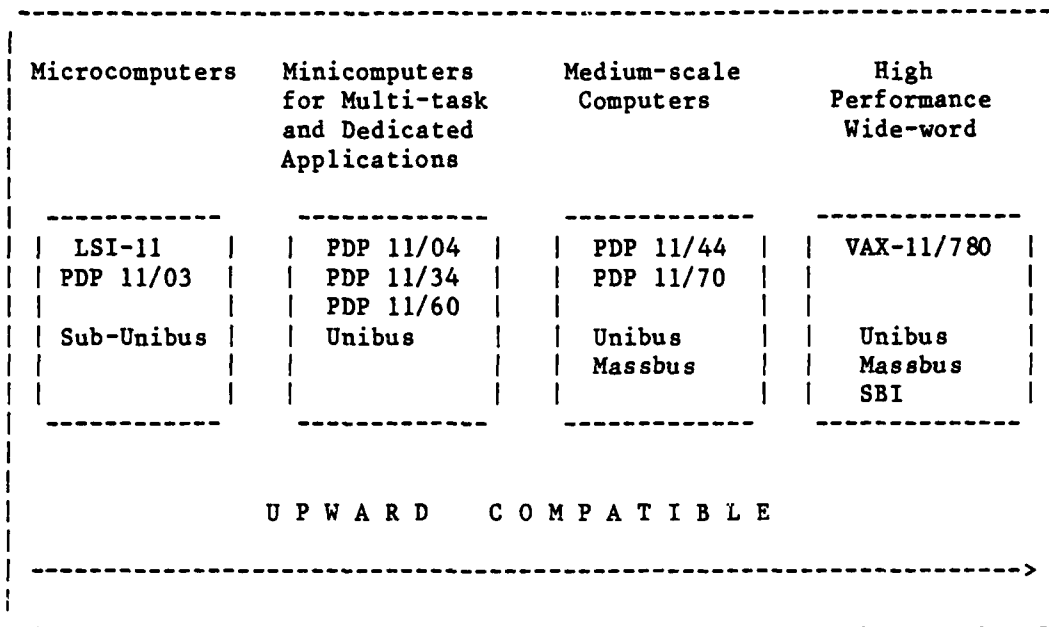


Figure 8. PDP-11 Processor Compatibility

consist of 56 signal lines, to which all devices, including the processor, are connected in parallel (Ref 9:11)(Ref figure 9). Communication between any two devices and the bus is in a master/slave relationship. During any bus operation, the bus master controls the bus when communicating with another device on the bus. When two or more devices try to control the bus at once, a priority scheme is used to decide among them. The bus is always given to the requesting device with the highest priority. If two devices of equal priority request control of the bus, the device that is electrically closer to the processor is given control. The priority arbitration takes place asynchronously, in parallel with data transfer. Every device on the bus except memory is capable of becoming a bus master (Ref 9:13).

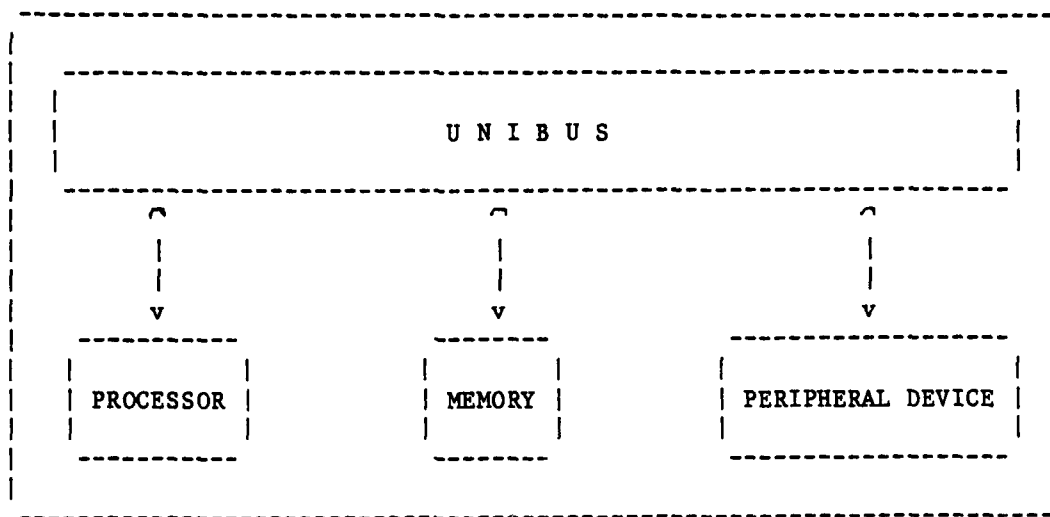


Figure 9. Unibus Configuration

The unibus architecture and the priority scheme is further defined in Figure 9.

Another feature that the PDP 11/34 and PDP 11/70 have in common is the RSX-11M operating system. RSX-11M is a small to moderate-sized real-time multiprogramming system that can be generated for a wide range of application environments, from small, dedicated systems to large, multipurpose real-time application and program development systems. RSX-11M makes it transparent to the user whether the PDP 11/34 or the PDP 11/70 is the host computer. The operating system is sophisticated enough to simultaneously support many users with little reduction in response time and also satisfy the critical time response real-time programs require. Software generated under the RSX-11M operating system can be executed on the VAX-11/780 in the "compatibility mode". The "compatibility mode" allows the VAX-11/780 to simulate the PDP-11.

Another feature in common between the PDP 11/34 and the PDP 11/70 is the means by which they manage their memory. A 16 bit virtual address provides direct access to 28K words (K=1024) of main memory. A program executed on the PDP 11/34 and 11/70 processors with memory management can address up to 32 K words of main memory by mapping all of the virtual address space to physical memory (Ref 9:147). The memory management system on the PDP 11/34 can map a 32 K program into any of the 124 K words of main memory. The memory management system on the PDP 11/70 allows a 32 K program to be mapped into any of the 1920 K of main memory. The key attributes of memory management are (1) it extends memory address space and provides protection and relocation features for multiuser applications.

PDP 11/34. The PDP 11/34 is a general purpose computer manufactured by the Digital Equipment Corporation for multi-task and dedicated applications. It contains hardware multiply/divide instructions, memory management, and enhanced data paths, and control signals for the addition of hardware floating point and cache memory options (Ref 9:181). The features common to all PDP 11/34 processors are (Ref 9)

1. Self-test diagnostics routines which automatically executed every time the processor is powered up.
2. An operator front panel with built-in CPU console emulator allows control from any ASCII terminal without the need for the conventional front panel with display switches and lights.

3. Automatic bootstrap loader
4. Up to 124 K words of either Core or MOS memory
5. Memory management

The options available for the PDP 11/34 processor are (Ref 9)

1. Integral extended instruction set that provides hardware fixed-point arithmetic in double precision mode.
2. Hardware floating point option allow ten times the performance of software implementations (Ref 9:182).
3. Cache memory option mean up to 60 percent system performance improvement (Ref 9:182).
4. RSX-11M operating system
5. A host of compilers and assemblers which include Pascal and FORTRAN.

The sponsor's PDP 11/34 configuration is shown in Figure 10.

PDP 11/70. The PDP 11/70 is designed to operate in large, sophisticated, high-performance systems. The central processor performs all arithmetic and logical operation required in the system. Memory management is standard with the PDP 11/70, allowing access up to two million words of memory (Ref 9:277). The cache contains 1024 words of fast memory (Ref 9:301). Optional high-speed, mass storage controllers

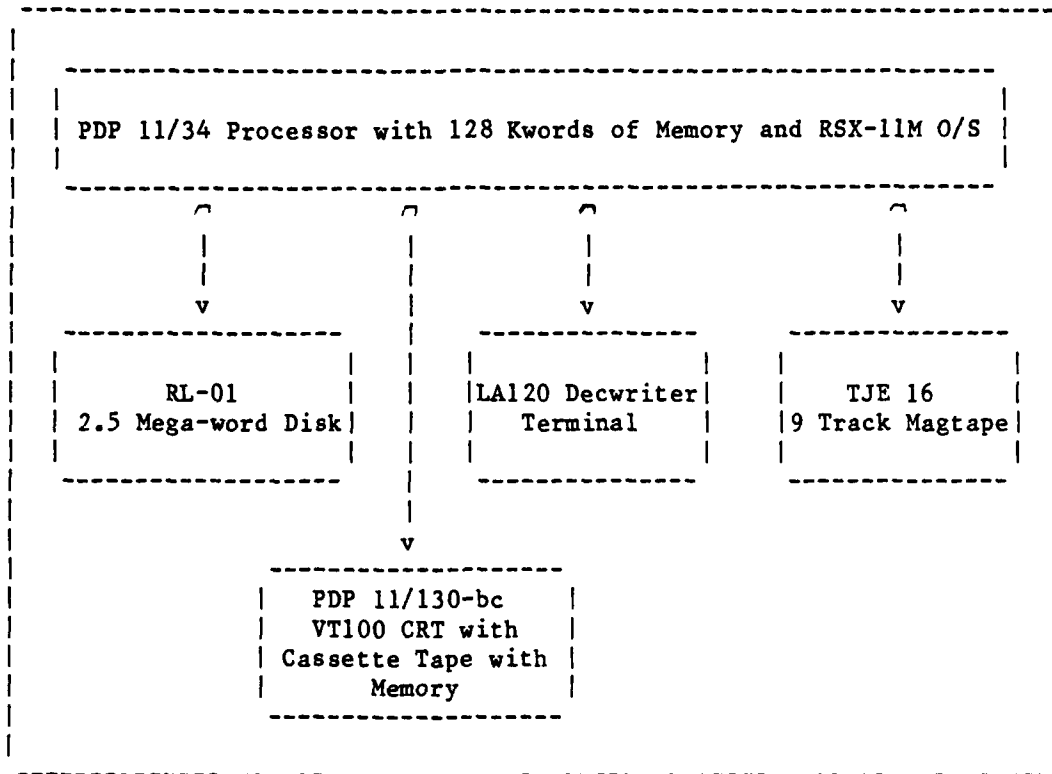


Figure 10. Sponsor's PDP 11/34 Configuration

provide dedicated paths to high performance storage devices (Ref 9:312-313). The PDP 11/70 supports a host of operating systems and compilers including the RSX-11M operating system and the Pascal compiler. The sponsors PDP 11/70 configuration is shown in Figure 11.

VAX-11/780. The short description of the VAX that follows is not included because the VAX is a part of CGS, but because it is the computer that will be used to develop CGS.

The VAX-11/780 is the virtual address extension of the PDP-11 family of computers. It is a 32 bit system that uses a very complex

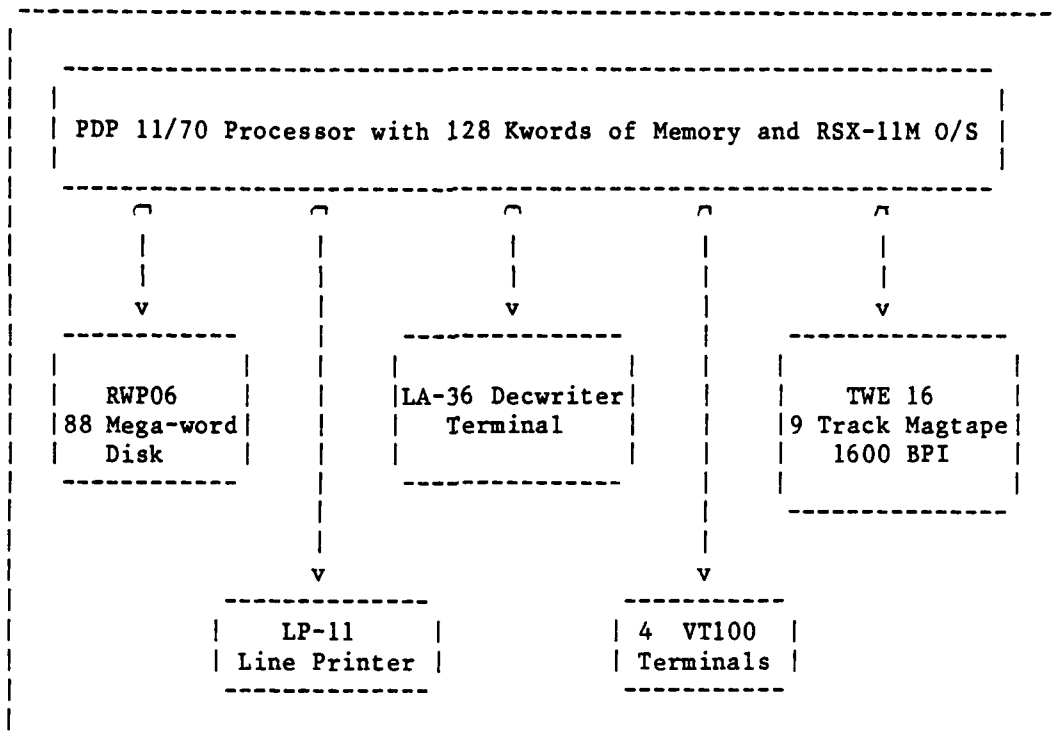


Figure 11. Sponsor's PDP 11/70 Configuration

virtual operating system. The 32 bit word size allows the VAX to have up to two gigawords of unique address space (Ref 11:345), and the virtual operating system removes all constraints on program size (Ref 10:255). Even though the VAX is much larger than the PDP-11 processors in both size and capacity, it still has the PDP-11's same basic structure, consisting of the high speed memory, the CPU, and the I/O system (Ref 10:5). The VAX does not support the RSX-11M operating system directly, but it does provide a "compatibility mode" which allow the VAX to execute programs developed on the PDP-11 computers with no modifications. DEC has made it very easy to develop programs on the VAX

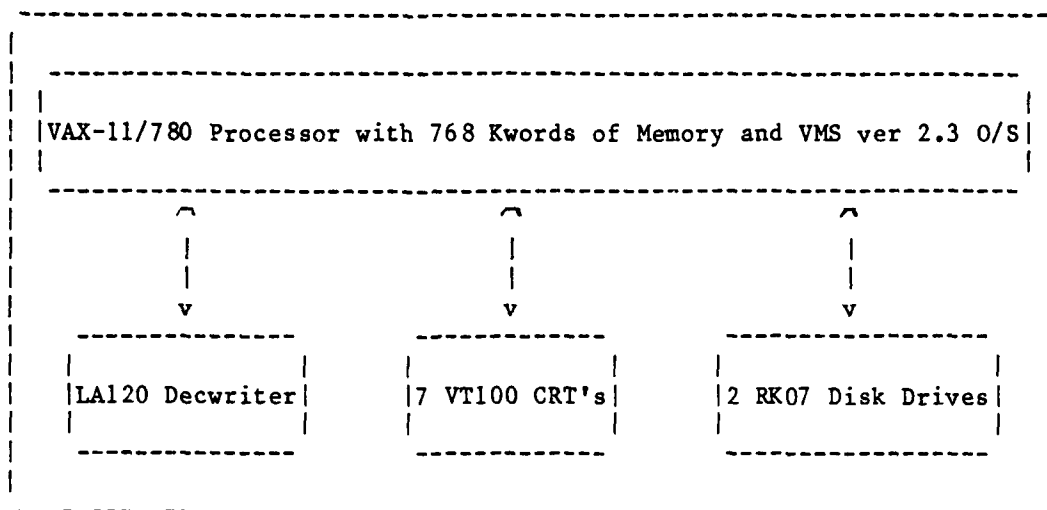


Figure 12. AFIT VAX-11/780 Configuration

that will later be executed on one of the PDP-11 family of computers. A detailed description of the VAX can be found in references 10 and 11. The configuration of the AFIT digital engineering Laboratory's VAX-11/780 is shown in Figure 12.

Summary

In summary, this chapter was devoted to the analysis of the CGS requirements. First, the results of the user survey were described followed by a description of a proposed system to satisfy the requirements and constraints. Next, the hardware components of this system were described. The major components in the system are: the ALR-46, the VT100 CRT, the Tektronix 4027 CRT, the hardware monitor, the threat simulator, the PDP 11/34, the PDP 11/70, and the VAX-11/780.

The constraints imposed by the sponsor on CGS hardware compatibility and availability did limit the hardware that could be used in CGS. Even though system requirements were discussed in this chapter, the primary emphasis was on hardware requirements. The software requirements for CGS can be found in the next chapter.

III. Software Requirements

Introduction

The previous chapter described the CGS hardware requirements/design in detail, but only briefly described the CGS software system requirements. This chapter further defines the software requirements of CGS. The software requirements define the type of communications between the user and CGS, as well as the output capabilities of CGS. The general structure of CGS is described, followed by a description of the communications link between the user and CGS. The requirements of both the Parameter and the RWR modes of operation are discussed. Finally, the language characteristics required to support the operational capabilities of CGS are described.

Software Overview

The CGS can be divided into three major categories : (1) Operator Interface, (2) Processing, and (3) Output. As shown in Figure 13, the user is directly affected by only two of the three categories (operator interface and output). The operator interface provides the user with a means of communicating with CGS. The user can request help or request one of CGS's display processes be executed. Even though the operator interface and the output processes used the same device, they are separate functions. The operator interface provides the user with a means of requesting CGS perform specific functions while the output processes provides a means of displaying the results of the user's

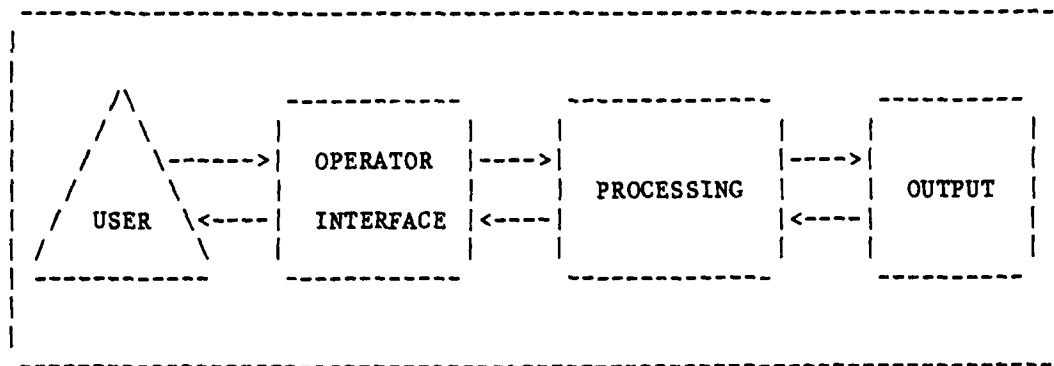


Figure 13. Man / ALR-46 CGS Interface

request. For example, when the user request the RWR mode, the operator interface receives the user command and passes it to the processing function. The processing function generates the display data and passes it to the output function, which displays the requested information. The user is concerned only with whether or not the display data was generated, not with how the processing function generated the display data. However, the user can be indirectly affected by the processing. For example, if an error occurs in the processing phase, the user would be notified via the operator interface. The operator interface is the only means of communications between the user and CGS.

Operator Interface

This section describes the characteristics of a model operator interface. The operator interface is the user's means of controlling the system. Unfortunately, most interfaces are not very efficient at communicating with their users (Ref 12:19). They often fail to understand what their users want them to do and then in turn, are unable to explain the nature of the misunderstanding to the user. Most

interactive computer systems respond only to commands phrased with total accuracy in a highly restricted artificial language designed specifically for that system (Ref 12). If the user fails to use this language or makes even the slightest mistake, an error message is issued followed by a request to reenter the command. This is in strong contrast to communication between people, which is conducted through a wide-ranging, natural language and is conducted despite many errors in language use. If a speaker or writer makes a syntactic error, his listener or reader does not normally fail to understand, but instead makes the obvious correction. The inability of current interactive systems to make such corrections is very frustrating for their human users.

Among other contributors to the man-machine communication barrier is the poor performance of interactive systems in keeping track of dialogue context and using it to resolve elliptical or anaphoric references people often use for economy of phrasing (Ref 12:19). Again, people, but not computers, are adept at suspending one context temporarily, switching conversation to a different context, and then popping back to the original one (Ref 12,15). A user who enters an illegal command while using a typical operator interface, is forced to seek help and then reenter the entire command. The error may have been as trivial as a misspelled word, but the interface does not care. It is very unforgiving! The operator may have several errors in the command, and yet the interface forces each error to be resolved one at a time, since it stops interpreting the command when it encounters an error.

The operator is not only forced to reenter the command several times, but is also prone to making new errors while trying to correct the old ones. Once a given mode (class of instructions such as edit, list directory, etc.) has been selected, the operator is locked in (Ref 15:90). If the operator needs to enter another mode to resolve a problem encountered in the current mode, the state of the current mode is lost. All of the problems described above can make an operator's first encounter with an interactive computer system very frustrating. Most of the problems normally encountered with an operator interface could be eliminated if the interface was more cooperative. In systems that do not use extremely abbreviated command languages, most spelling and syntax errors could be automatically corrected by comparing the command entered by the operator with the list of legal commands. In this way, the operator is only required to reenter the command when it is too different from any legal command to be recognized.

Research in human communication needs has resulted in a list of characteristics that an operator interface should possess (Ref 12):

1. Flexible parsing. Whenever people spontaneously use a language, whether natural or artificial, they make many small mistakes in syntax and spelling and frequently do not say exactly what they mean. A graceful system must be able to process its user's input when a linguistic error is made, and then to correct the mistakes if possible, or ask for a selection between alternative interpretations of what was said.

2. Robust communication. A graceful system must be able to let its user know when it understands and, more importantly, when it does not. While avoiding verbosity or an unnecessary disruption of the flow of conversation, it must keep the user informed of any assumptions it makes about what was said and allow opportunities to correct any misapprehensions that may have occurred. Conversely, and much more difficult, it should monitor the user's understanding of its output and try to correct any demonstrated misunderstandings.
3. Identification from description. A fundamental ability of any interface is that of recognizing objects known to it internally from a user's description of them. A graceful interface must also be able to negotiate with the user when the descriptions turn out to be ambiguous or when they have no referents. The interface should be able to do this without losing the larger context in which the description occurred.
4. Focus tracking. A graceful system should be able to track the focus of attention of the user as it change through the dialogue. It should be able to track it both within restricted contexts, such as occur in a description resolution, and across leaps, such as those that typically occur between separate commands to the interface. When such large leaps occur, it should also be prepared to follow the focus back to the original context. This allows the user to break off one command, execute another, and go back to the first. Keeping track of the focus is important in the resolution of elliptical and anaphoric inputs.

5. Natural output. A graceful interface should be able to produce output appropriate to the current context, with the correct amount of detail in the object descriptions it generates.
6. Explanation facility. A gracefully interacting system should be able to give explanations of both a static and dynamic nature to its user. Static explanations relate to what the system can and cannot do in a general sense, and how the user can ask the system to do a specific task. Dynamic explanations describe what the system is doing, why it is doing it, and the outcomes of past events.
7. Personalization. A graceful interface should recognize and adjust to the idiosyncracies and preferences of its user. This includes the ability to spot and correct recurring typographical, spelling, or syntactic errors. It should also maintain a model of the user's level of experience and adjust its messages and explanations accordingly.

Even though the design and implementation of an interactive system that fulfills the requirements outlined above is a very large task, systems such as Alto, Perq, Lisp (Ref 12) and Small Talk (Ref 15) prove it can be done. One of the main reasons that many current systems perform so poorly is that the implementors were unable or unwilling to expend so much effort on the user interface. The sponsor (Robins AFB) of this thesis feels the same as most system developers. The sponsor chose to have a less sophisticated operator interface and operational software developed verses a more sophisticated operator interface and no

operational software. The trade-off between a sophisticated operator and operational software was necessary because the limited time allowed to complete the thesis. A sophisticated operator interface would have taken several months to design and implement, which is more time than is allowed to complete the entire thesis. Of course, the primary function of the operator interface is to provide the user with the Parameter and RWR modes. They are discussed in the following sections.

Parameter Capability

As mentioned in Chapter II, the Parameter capability provides the users with a means of monitoring the critical, OFP track file parameters (Ref Table 2), without affecting the operation of the ALR-46 system. The functions required to provide the user with this capability are described below:

1. The Parameter display must be protected against itself. That is, the operator / computer dialogue must be kept in a dedicated area of the display and the same is true for the parameter output. The results of either of these outputs infringing on the area of the other are obviously catastrophic.
2. During the Parameter sequence, the critical parameters on the ALR-46 system display (Ref figure 14) must be identified. Of course, the length of the names used on the actual display will be determined by the space available.

BY PRIORITY	Threat 1 Data	Threat 16 Data
SYMBOL	:	:
TYPE	:	:
BANDS	:	:
PRI / FRAME PERIOD	:	:
PULSE TRAIN DESCRIPTION	:	:
PRI AVERAGE	:	:
DEVIATION	:	:
SPECIAL	:	:
POWER /PWR ROUND	:	:
HIGH BAND SINE	:	:
HIGH BAND COSINE	:	:
CEPC	:	:
HI BAND AGE COUNT	:	:
CD BAND AGE COUNT	:	:
PRIORITY	:	:
PRIORITY / ROUND	:	:
AZIMUTH	:	:
LETHALITY RING	:	:
ML AGE	:	:
RECORD NUMBER	Threat 1 Data	Threat 16 Data
OPERATOR / COMPUTER DIALOGUE REGION		

Figure 14. Parameter Mode User Display

- Another critical function in the Parameter mode is the generation and maintenance of the user's display. In this mode, all threat data collected by the hardware monitor is decoded and displayed for up to sixteen threats simultaneously. Each time a threat parameter in the track file is changed, the hardware monitor sends a copy of the data to CGS. CGS decodes the word and updates the critical parameters of the appropriate threat. For a detailed description of the data collection and decoding processes, refer to the "Parameter / RWR Data" section. A detail description of the critical

parameters to be displayed can be found in Appendix A. Another useful function provided by CGS is the RWR capability described in the following section.

RWR Capability

As mentioned in Chapter II, the RWR capability provides the user with a means of simulating the RWR display in color as well as providing range and azimuth data. The functions required to provide the user with the capability are as follows:

1. As with the Parameter display, the RWR display must be protected against itself. That is, the operator / computer dialogue must be kept in a dedicated area of the display and the same is true for the graphics output. The results of either of these outputs infringing on the area of the other is also extremely detrimental.
2. During the RWR sequence, parts of the pilot's display (Ref figure 15) are static, which means it only has to be drawn once. The static parts of the display are:
 - (a) Range rings. The display has three concentric circles (rings) which represent distance. For example, if the display were defined at sixty miles maximum, the outside ring would represent sixty miles, the next ring would represent forty, etc. A small dot, which represents the position of the aircraft, must be placed in the center of the circles.

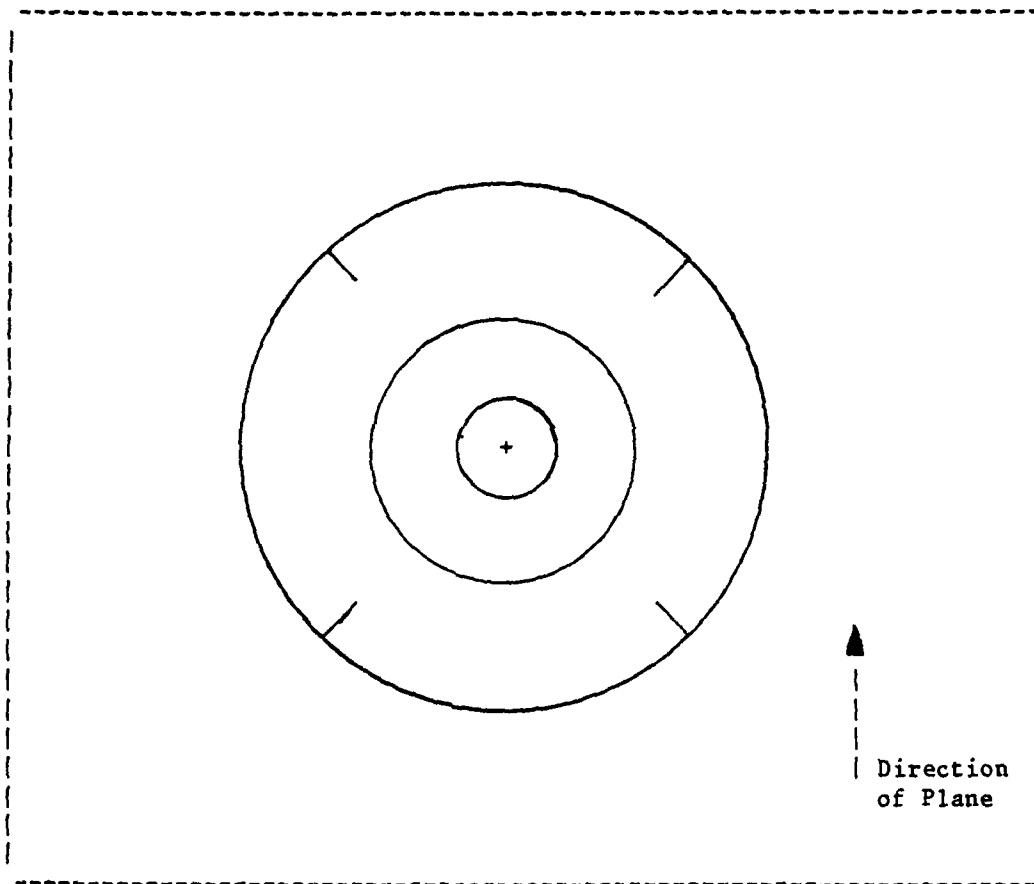


Figure 15. Static ALR-46 Display

- (b) Azimuth Indicators. The circles are divided into four equal quadrants by a vertical and horizontal line. The two lines intersect in the center of the circles and extend to the outer most circle. The upper half of the vertical line represents zero degrees azimuth. The azimuth angle increases in the counter-clockwise direction. Four hash marks start on the outside ring and extend inward toward the center of the circles

for approximately one-half inch. The hash marks are positioned at azimuth angles: 45, 135, 225, and 315 degrees.

3. The RWR mode user's display (Ref figure 16) must be generated and maintained. If a change was made to a threat's symbol, range, or azimuth parameters, it must be incorporated into the RWR mode user's display. This RWR user's display is divided into three regions. Region one, the operator / computer dialogue region, is used to display messages from the computer and to enter user commands to the CGS. It is not affected by a threat parameter change. Region two is used to display a limited number of critical threat parameters. Specifically, the threat symbol code, range, and azimuth. This region is updated each time the threat's symbol, azimuth, or range changes. The pilot's display region, or region three, is a replica of the actual pilot's display except for the addition of color. When a threat is first recognized by the ALR-46 system, its symbol is displayed at the appropriate location (based on its range and azimuth) using a unique color. Each time a change in range or azimuth occurs, the threat symbol must be repositioned accordingly. When the threat is no longer active, its symbol is removed from the screen. If the threat's range extends beyond the maximum display range, then the threat symbol is not displayed. Of course, the integrity of the display is entirely dependent on the ability of CGS to decode the information taken from the OFF by the hardware monitor. The decoding process is described in the following section.

EMITTER NUMBER	SYMBOL	RANGE	AZIMUTH	
1	SA2	20	310	
2	AAA	50	48	
:	:	:	:	ALR-46 Pilot's Display
:	:	:	:	(Reference Figure 4)
:	:	:	:	
:	:	:	:	
:	:	:	:	
:	:	:	:	
:	:	:	:	
:	:	:	:	
:	:	:	:	
:	:	:	:	
:	:	:	:	
16	SA8	100	165	
(REGION # 2)				(REGION # 3)
OPERATOR / COMPUTER DIALOGUE REGION				(REGION # 1)
ALR-46 CGS>				

Figure 16 RWR Mode User's Display

note: Region 3 is not drawn to scale.

Parameter / RWR Data

A detailed description of the ALR-46 operation flight program's (OFF) emitter track file will help the reader understand what is

required to generate the data needed to drive the Parameter and RWR modes user's display. Actually, the emitter track file is not a file, but a buffer within the OFP. The OFP, which resides in the memory of the ALR-46 Flight Processor, maintains a set of pointers (first_available_record and first_emitter) which point to records in the track file (Ref figure 17). If the first_available_record points to itself, it means no records are available. If the first_emitter pointer points to itself, it means no emitters are active. The track files are divided into sixteen records, which contain sixteen words each. As shown in Figure 17, the records in the track file form two linked lists, the "Red_Link_List" and the "Blue_Link_List". The Red_Link_List is composed of records of active emitters while the Blue_Link_List contains all of the records that are available for use. The first_emitter_pointer is the beginning of the Red_Link_List. It points to the highest priority threat, the highest priority threat points to the next highest priority threat, etc., until the lowest priority threat is reached. The lowest priority threat points to the first_emitter_pointer. The first_available_emitter pointer is the beginning of the Blue_Link_List. It points to the first available record, the first available record points to the next available record, etc., until the last available record is reached. The last available record points to the first_available_emitter pointer.

When an enemy threat is recognized by the ALR-46, it is assigned, based on the threat's priority, a record in the track file. The pointers have to be updated in both linked lists, since an available

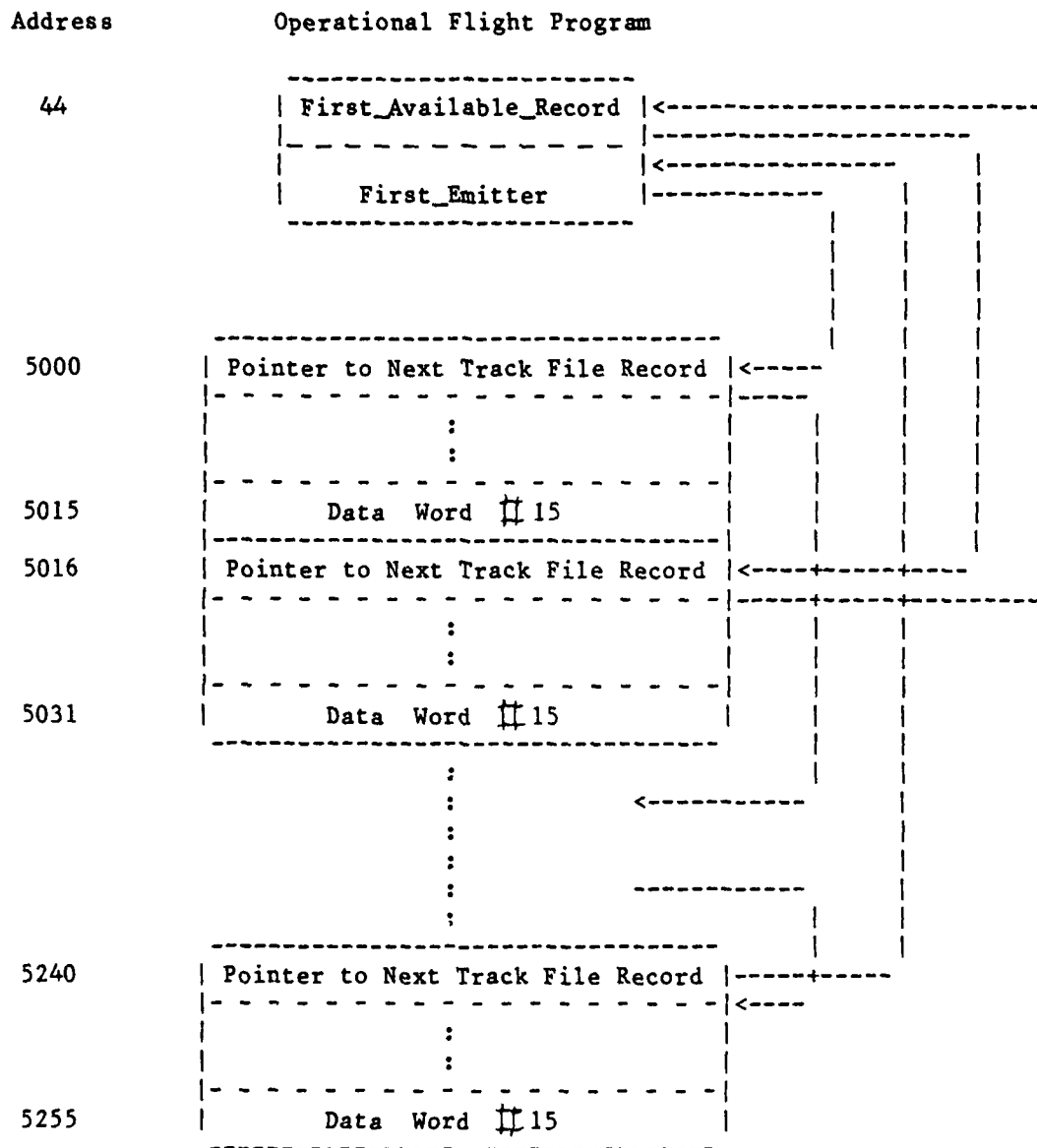


Figure 17. ALR-46 Emitter Track File

note: The pointer arrangement shown depicts fifteen used records (active emitters) and one available record.

record has to be removed from the "Blue_Link_List" and inserted into the "Red_Link_List". As the ALR-46 system collects more information about a threat (PRI, PW, etc.), the OFP updates the appropriate word(s) in the record associated with that threat. Each time a word in the emitter track file is updated, the hardware monitor stores the contents and the address of the updated word. The information needed to drive the RWR and Parameter displays must be extracted from the data (Ref Table 4) taken from the ALR-46 Flight Processor by the hardware monitor. The hardware monitor data will be in a sequential, fixed length record file. Each record will contain four thousand , sixteen bit words. As shown in table 4, the hardware monitor data will be either time data (in groups of four words) or threat data (in groups of two words). The definition of the bits in each of the sixteen words of a track file record is defined in Appendix A. The threat data is the only data required to drive the RWR display. Of course, the information on the display is only as good as the programs that drive the display. The integrity of the programs that drive the display are heavily influenced by the language they are written in, which brings us to the topic of the next section.

Selection of the CGS Language

The sponsor left the choice of whether to write the software in Pascal or FORTRAN up to the author. Pascal was chosen as the CGS language for several reasons:

1. One of the major software problems through-out industry and the

government today is the high cost of maintaining software. This cost could be reduced if the software was better documented and structured. Pascal is a structured language with many structured commands, such as "while", "repeat until", and "if then", etc. (Ref 2). A structured program is normally easier to understand and thus easier to maintain.

2. Unlike FORTRAN, Pascal does not limit the variable names to six or nine characters as FORTRAN does (Ref 3). More meaningful variable names make the program more readable and easier to understand.
3. Pascal allows the declaration of record structures which make the program more readable (Ref 2).
4. Pascal supports linked list (Ref 2). Using linked lists with FORTRAN is very awkward because FORTRAN does not support the function.

The only advantage of FORTRAN over Pascal is the fact that more of the sponsor's personnel were already familiar with FORTRAN than Pascal. Since the benefits of using Pascal out-numbered the benefits of FORTRAN, Pascal was chosen as the programming language of CGS.

Summary

To summarize, this chapter has been devoted to discussing the CGS software requirements and selecting a programming language for CGS. A brief summary of the software requirements are as follows:

1. The operator interface must be friendly, clear, and forgiving. It must lead the inexperienced user "by the hand" with descriptive prompts, and provide a very short command language for the experienced user. It must also provide help to the users when necessary.
2. The Parameter capability requires the following functions to be performed:
 - (a) The different regions in the user's display must be protected from each other.
 - (b) The static regions of the Parameter display have to be drawn.
 - (c) The data taken from the OFF by the hardware monitor have to be decoded.
 - (d) The dynamic regions of the Parameter display have to be created and maintained.
3. The RWR capability requires several different functions to be performed:
 - (a) The different regions in the user's display must be protected from each other.
 - (b) The static regions of the RWR display have to be drawn.
 - (c) The data taken from the OFF by the hardware monitor must be decoded.
 - (d) The dynamic regions of the RWR display must be created and maintained.

This chapter completes the definition of the ALR-46 CGS requirements. The transformation of the CGS system requirements into an operational system is discussed in Chapter IV.

IV. Design of Computer Graphics System

Introduction

The previous chapter employed the user survey to determine the software system requirements. In this chapter, the software requirements along with supporting background information were used to design CGS. In the first phase, Structured Analysis (Ref 8) was employed to build a logical model of CGS. Transform and Transaction Analysis techniques (Ref 17) were used in the final phase of the design to convert the logical model into a physical model. The module structure charts created by the transform and transaction analysis process, are found in Appendix D.

Logical Model

Introduction. While the software requirements were discussed in Chapter III, further analysis could prevent errors from being designed into CGS. Several techniques were available for this task including Weinberg's Structured Analysis (Ref 8), Softech's Structured Analysis and Design Technique (SADT)(Ref 16), and IBM's Hierarchical Input-Process-Output (HIPO) diagrams (Ref 8). Structured Analysis was selected to build the logical model because it offers several advantages. However, before the advantages can be discussed, it is necessary to understand the basic components of the Structured Analysis technique.

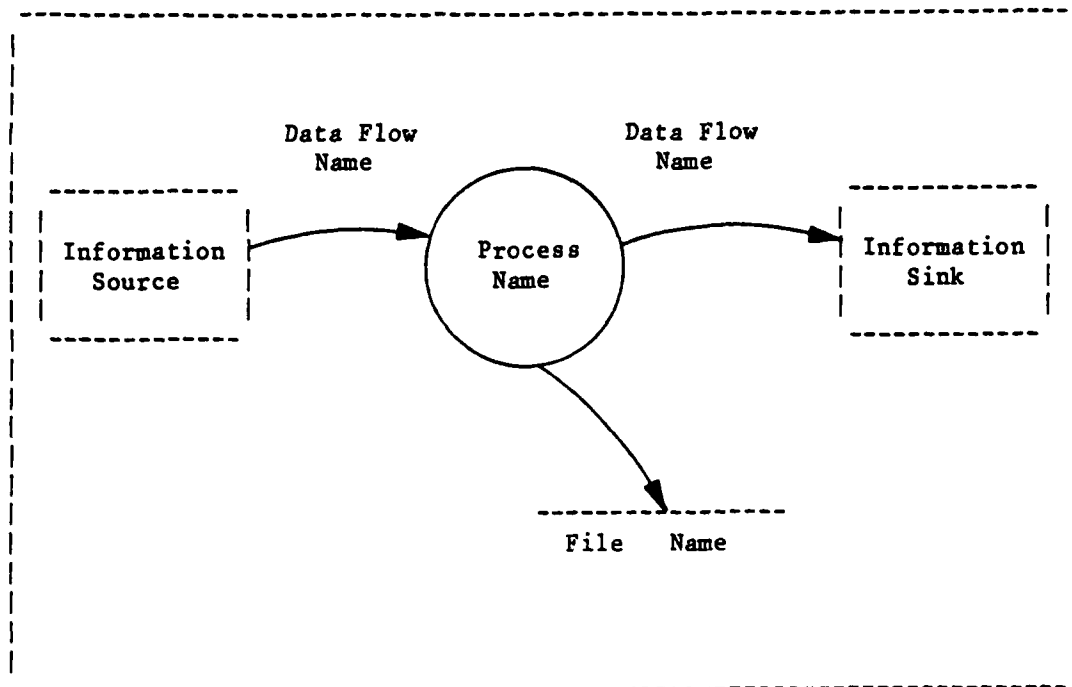


Figure 18. DFD Components

Structured Analysis uses data flow diagrams (DFD's) and a data dictionary to logically define the system requirements. The DFD's have the four components shown in Figure 18. The first component is the data flow, a "pipeline" of other data flows and of data elements. The data elements are basic data types that can not be partitioned further and still retain their meaning. For example, the data element might be an integer word representing the number of active threats at a specific instant in time. The data flow is represented by a curved arrow on the DFD's. The process converts input data flows to output flows and is the second component of the DFD's. It is represented by a circle that contains the process name. The box represents the third component of

Table 5. DFD Index

	PAGE
DFD Components	62
Context Diagram	64
System Diagram	65
Display Critical Track File Parameters (2.0) DFD	67
Update CGS's Track File Entry (2.2) DFD	68
Simulate Pilot's Display (3.0) DFD	69
Extract / Format RWR Data (3.3) DFD	71
Help User (4.0) DFD	72

DFD's, the sources and sinks of information. They may represent a user keyboard, a user display, or any other mechanism by which information enters or leaves the system. Finally, files are the last component and are repositories of information within the system. They are shown by straight lines. The DFD's are layered, starting with a context diagram that defines the interface of the system with its environment. Then the processes in the diagram are expanded into lower level DFD's. This partitioning process continues until each process has been defined in sufficient detail, or in terms of its most elementary inputs and outputs (Ref 8:78).

With this very abbreviated description of Structured Analysis, it is now possible to examine the three primary advantages Structured Analysis offered. First, it was based on the concept of partitioning. This allowed each CGS requirements to be analyzed in an orderly manner. Second, Structured Analysis differentiated between the logical and the physical environment, thus more clearly defining the problem.

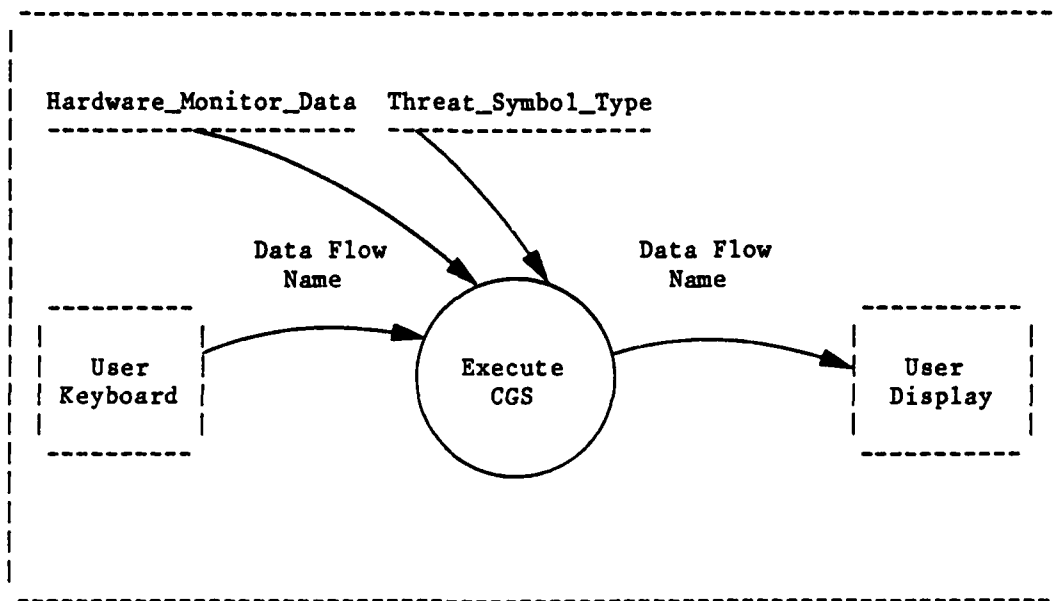


Figure 19. Context Diagram

Finally, it was fairly easy to transition from the DFD's in the logical model (Ref Table 5) to the module structure charts (Ref 17) used in the physical model.

Context Diagram. The context diagram shows the system boundary and interface with the user. As shown in Figure 19, CGS must include everything from the user input at the keyboard to the response the user receives at the display.

System Diagram. The system diagram translated the software requirements for CGS into a logical model as shown in Figure 20. First, the user command must be examined to determine the type of user request (process 1). If it is a Parameter command, CGS must activate the

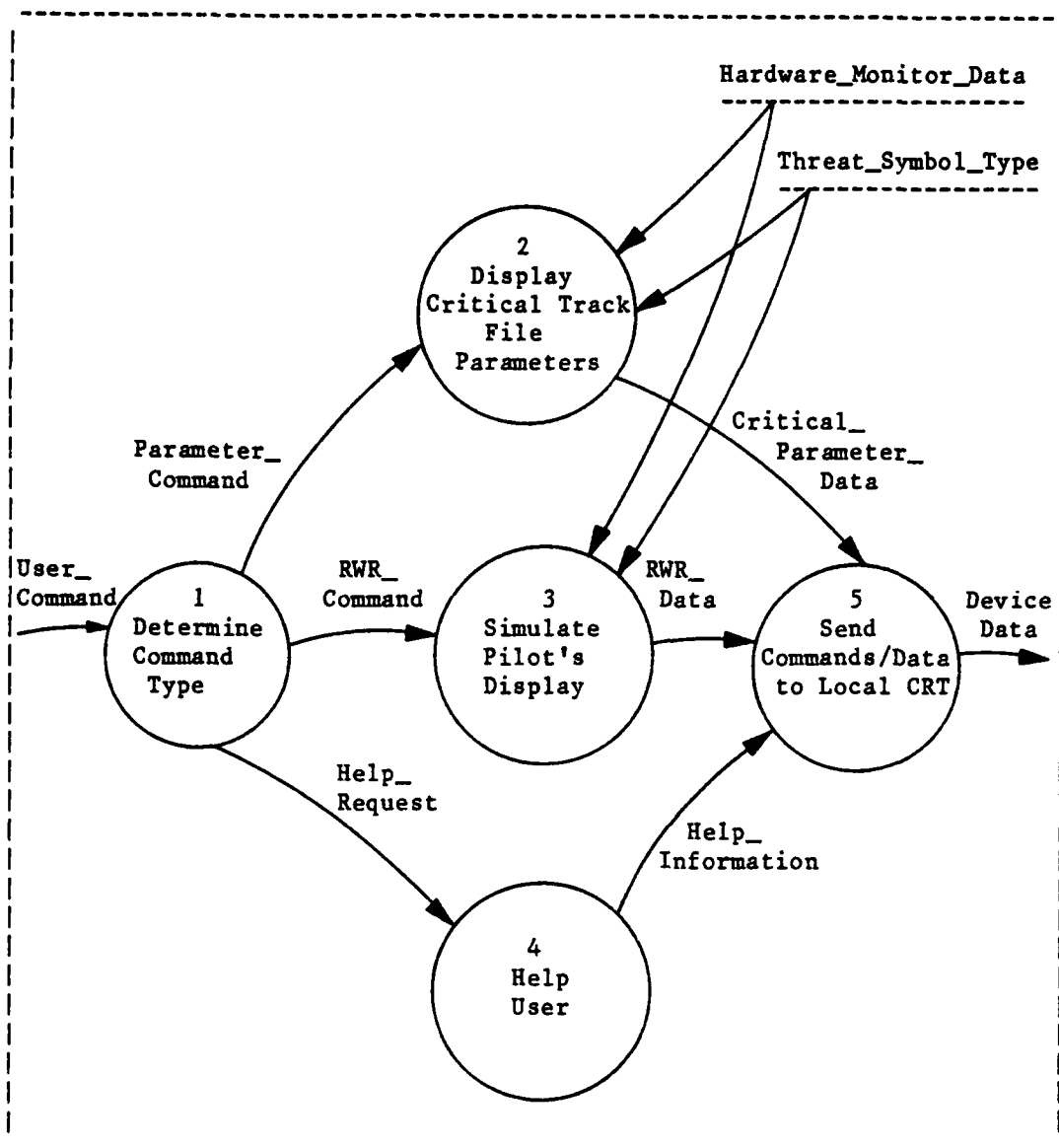


Figure 20. CGS System Diagram

process to display the critical track file parameters (process 2). Both the hardware monitor data file and threat symbol type file must be accessed to generate the critical parameter data. If the user command

is an RWR command, CGS must activate the process to simulate the pilot's display (process 3). Again, both the hardware monitor data and the threat symbol type files must be accessed by the process to generate the RWR data. If the user command is a request for help, the appropriate help information must be output to the user (process 4). Finally, the response output from one of the three processes must be transmitted back to the user (process 5).

Display Critical Track File Parameters. The display parameters process was specified in more detail by the lower level DFD shown in Figure 21. First, a block of data (4000 words) must be read from the hardware monitor data files. The CGS must then extract an emitter track file data word from the block of data (process 2.1). Next, CGS must update its threat data base (process 2.2). In the final step, CGS must extract the updated parameters from the treat data base and prepare them for output to the user device (process 2.3).

Update CGS's Track File Entry. The update track file entry process may be specified further by the lower level DFD shown in Figure 22. First, the specific threat being updated must be indentified (process 2.2.1). This can be done using the address of the data words (Ref figure 17). After the threat has been identified, CGS must identify which of the sixteen possible word (Ref Appendix B) has been modified (process 2.2.2). Finally, the critical parameters affected by the change must be updated (process 2.2.3). As shown in Appendix A, several critical parameters can be affected when only one track file word is changed.

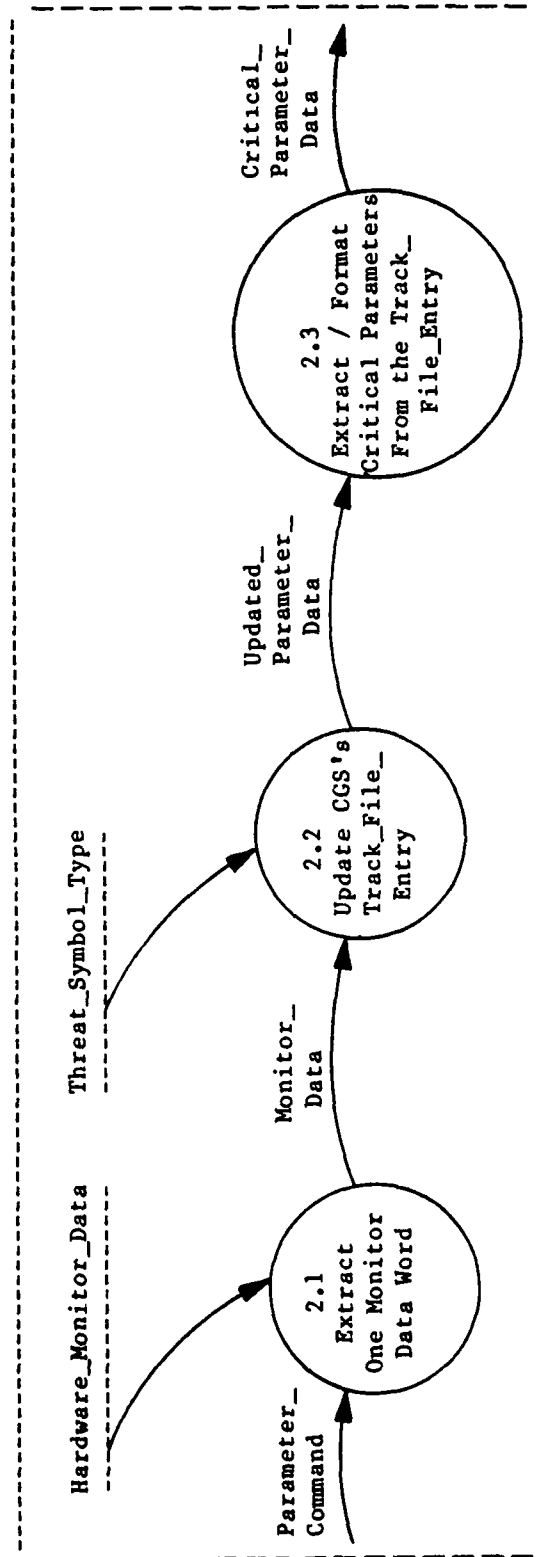


Figure 21. Display Critical Track File Parameters (2.0) DFD

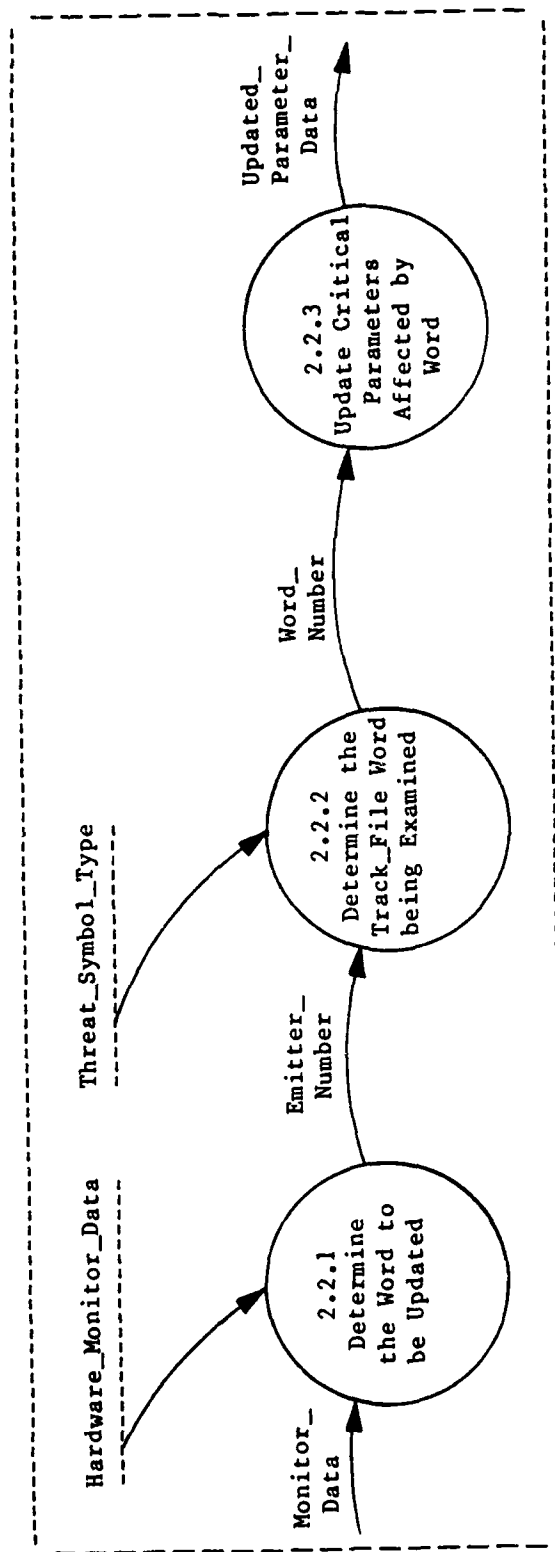


Figure 22. Update CGS's Track File Entry (2.2) DFD

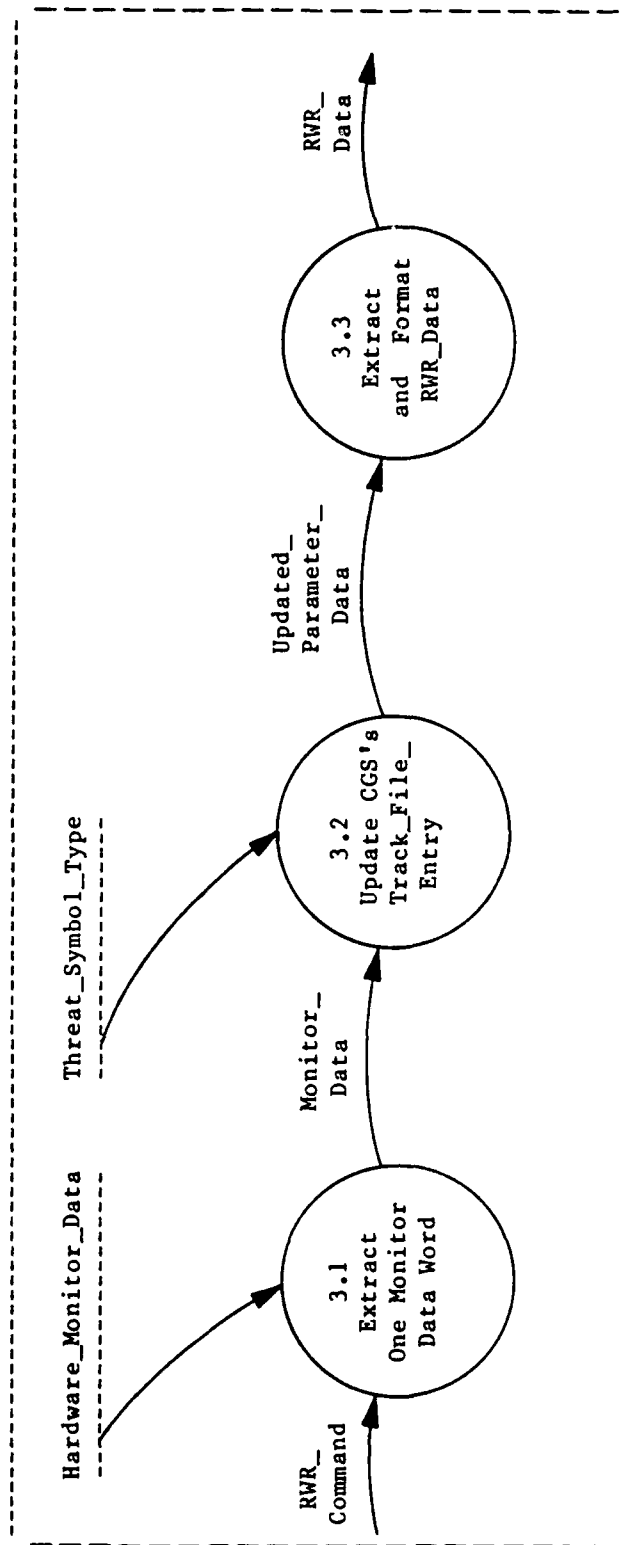


Figure 23. Simulate Pilot's Display (3.0) DFD

Simulate Pilot's Display. The simulate pilot's display process was specified in more detail by the DFD shown in Figure 23. First, a block of data (4000 words) must be read from the hardware monitor data file. The CGS must then extract an emitter track file data word from the block of data (process 3.1). Next, CGS must update its threat data base (process 3.2). (Both the extract data word (process 3.1) and the update track file entry (process 3.2) processes are identical to processes 1.1 and 2.2 shown in Figure 21.) Finally, CGS must extract the updated parameters from the threat data base and prepare them for output to the user device (process 3.3).

Extract / Format RWR Data. The extract / format RWR data process was specified in more detail by the DFD shown in Figure 24. First, the position of the threat symbol must be determined (process 3.3.1). The position of the symbol is a function of the threat's range and azimuth. Next, the symbol must be removed from its current position on the display and displayed in its new position (process 3.3.2). The critical parameters on the user's display must also be updated (process 3.3.3). The threat's symbol, range, and azimuth are displayed alongside the simulation of the pilot's display. Finally, the RWR data is prepared for output by adding display position commands (process 3.3.4).

Help-User. Finally, the help-user process may be specified by the lower level DFD shown in Figure 25. The help request must be classified as either a general information request, an RWR display procedure request, or a Parameter display procedure request (process 4.1). If it is a general information request, a menu selection of the

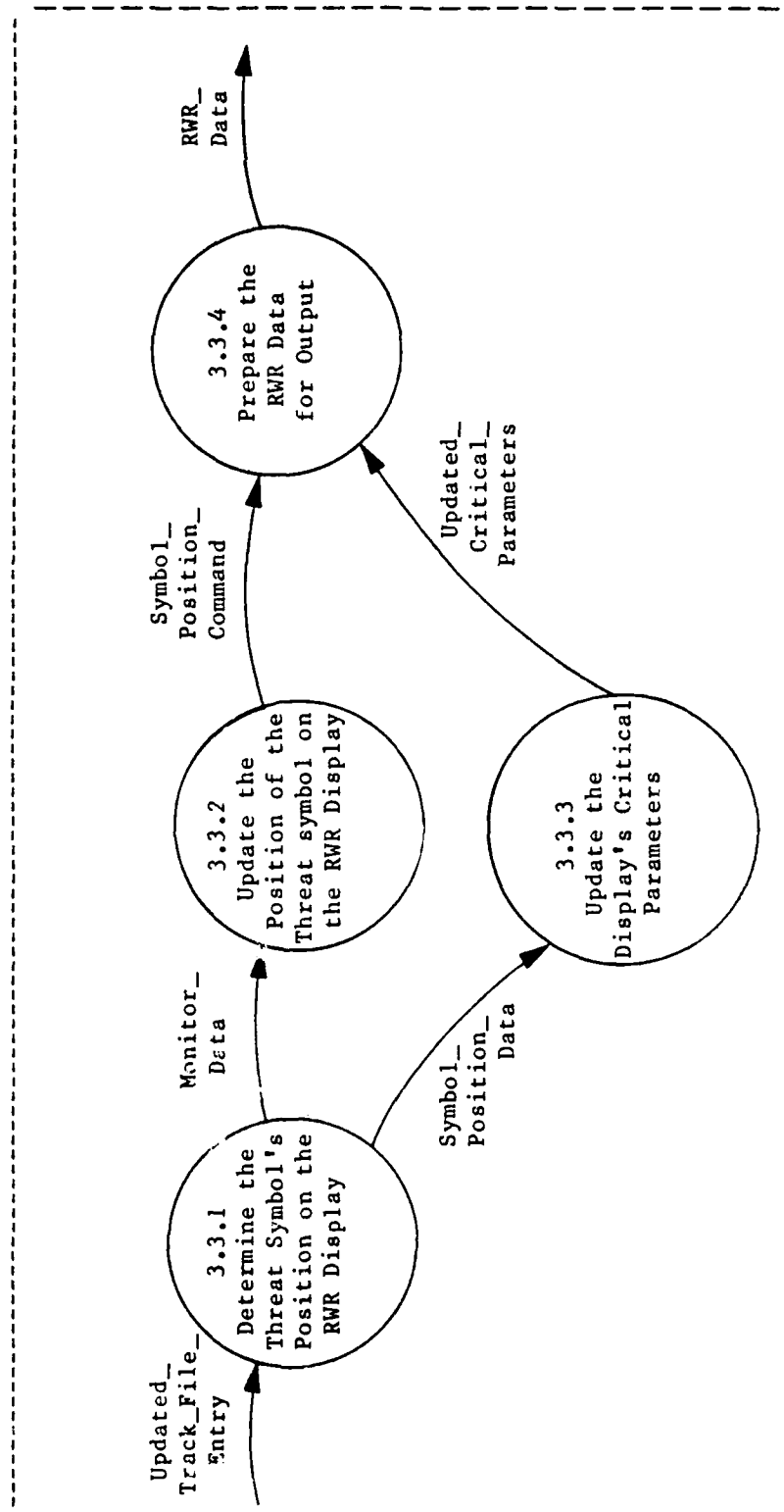


Figure 24. Extract / Format RWR Data (3.3) DFD

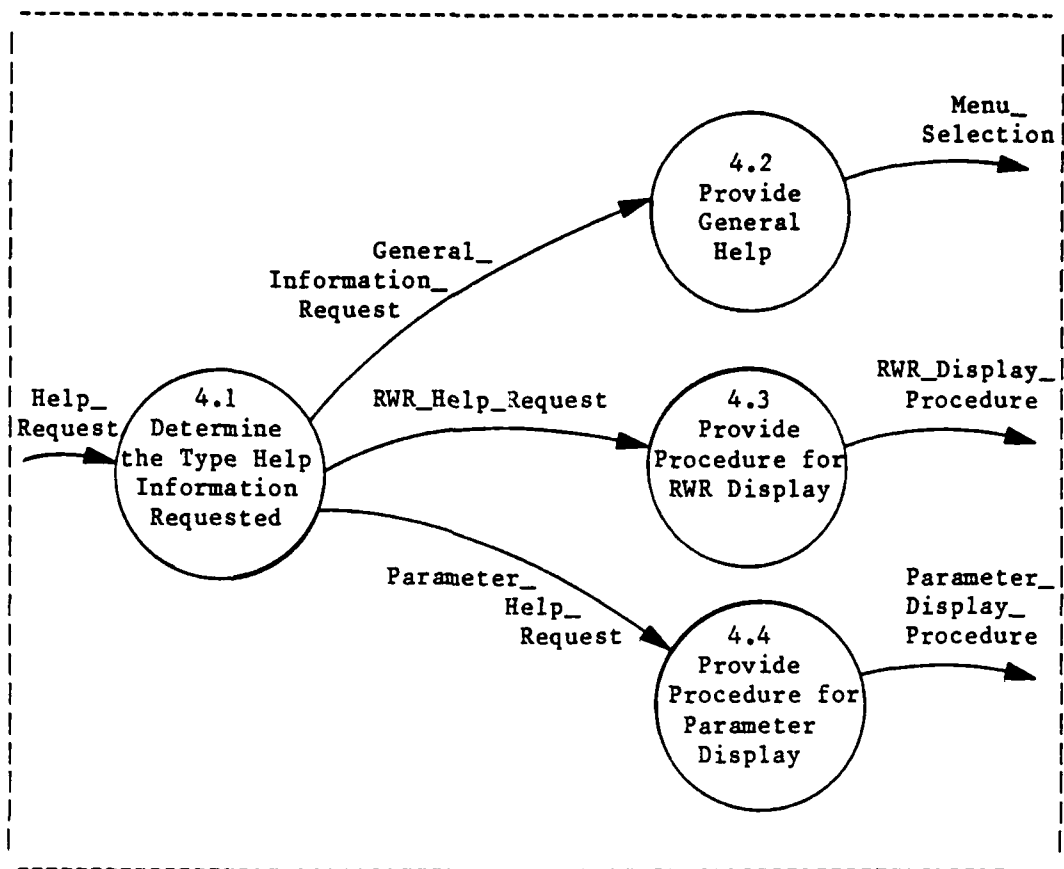


Figure 25. Help User (4.0) DFD

available commands and their formats must be output along with the formats of the more specific help requests (process 4.2). If the help request is a RWR display procedure request, the format for initiating the RWR procedure must be output (process 4.3). Finally, if the help request is a parameter display procedure request, the format for initiating the parameter procedure must be output (process 4.4).

The next phase in the design was to translate the logical model of CGS into a physical model. This process is described in the following section.

Physical Model

Introduction. The physical model was designed using structured design techniques proposed by Yourdon and Constantine (Ref 17). Two principal methods were used to translate the structured specification into module structure charts, transform analysis and transaction analysis. These techniques were chosen due to their direct correspondence to the DFD's specified in the previous section.

Transform analysis is used to translate a DFD into three sets of modules. The first set of modules gets data from the source. The data is transformed into the output by the second set, and the third set outputs the data to the sink. To partition the DFD into the three sets of modules, the DFD is divided into an afferent branch, a transform section, and an efferent branch. This is done by tracing the input from the source to the furthest data flow where it is still recognizable as an input. Likewise, the output is traced backwards from the sink to the first data flow where it is recognizable as an output. These two data flows then are used as the boundaries between the three sections of the DFD.

For the first set of modules, the structure is derived by making an afferent module for each data flow and a transform for each process. This is illustrated in Figure 26. For the second set of modules, the

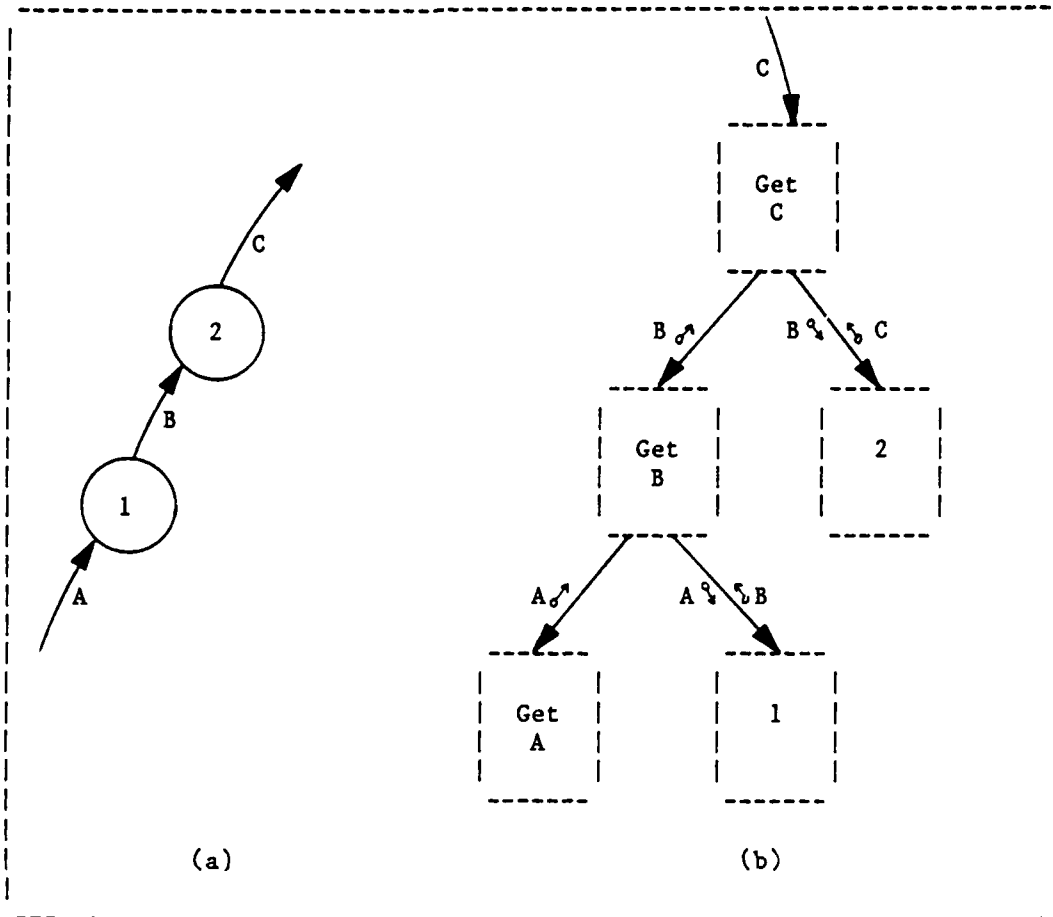


Figure 26. Factoring the Afferent Section

first module is factored into subordinate transform modules that perform the functions stated by the process names. An example of this is shown in Figure 27. The structure for the last set of modules is designed similarly to the set of afferent modules. As shown in Figure 28, the efferent module has subordinate to it another efferent module and a transform module that relates the two data flows specified by the efferent modules. A complete description of transform analysis is given

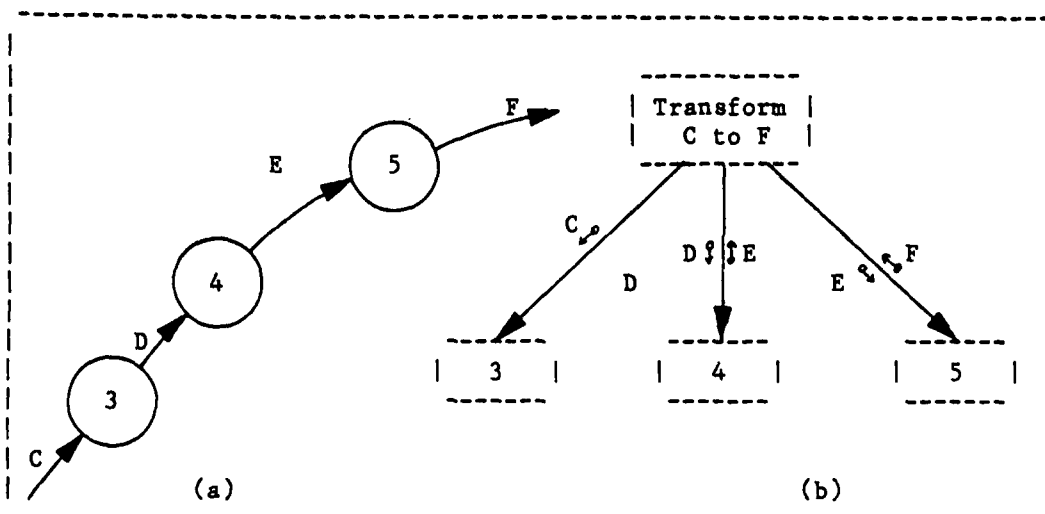


Figure 27. Factoring the Transform Section

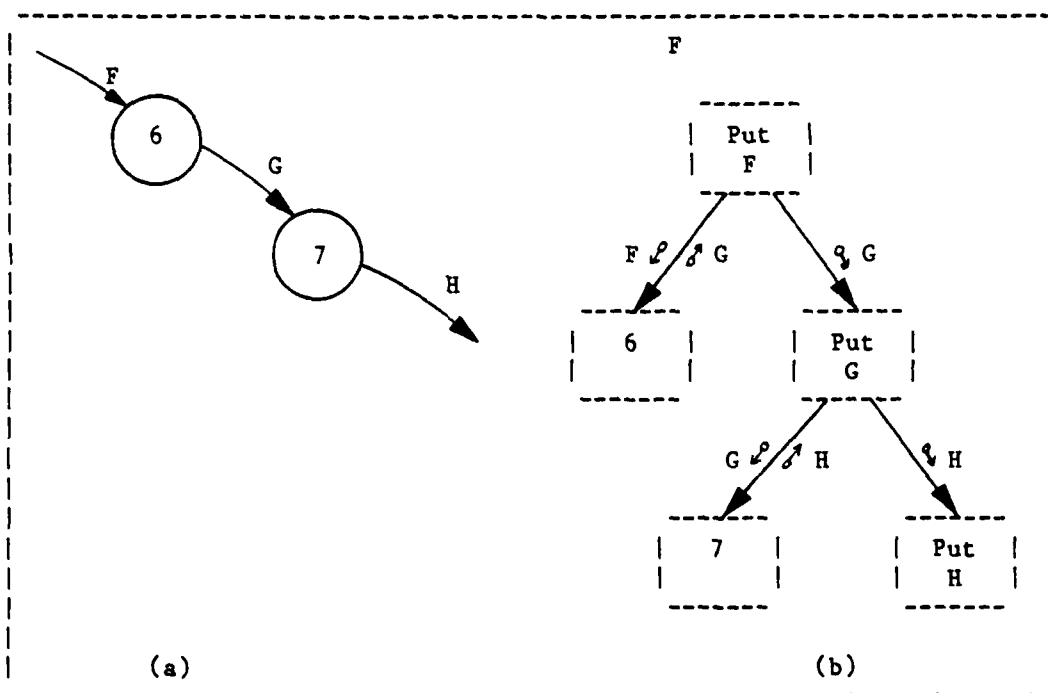


Figure 28. Factoring the Efferent Branch

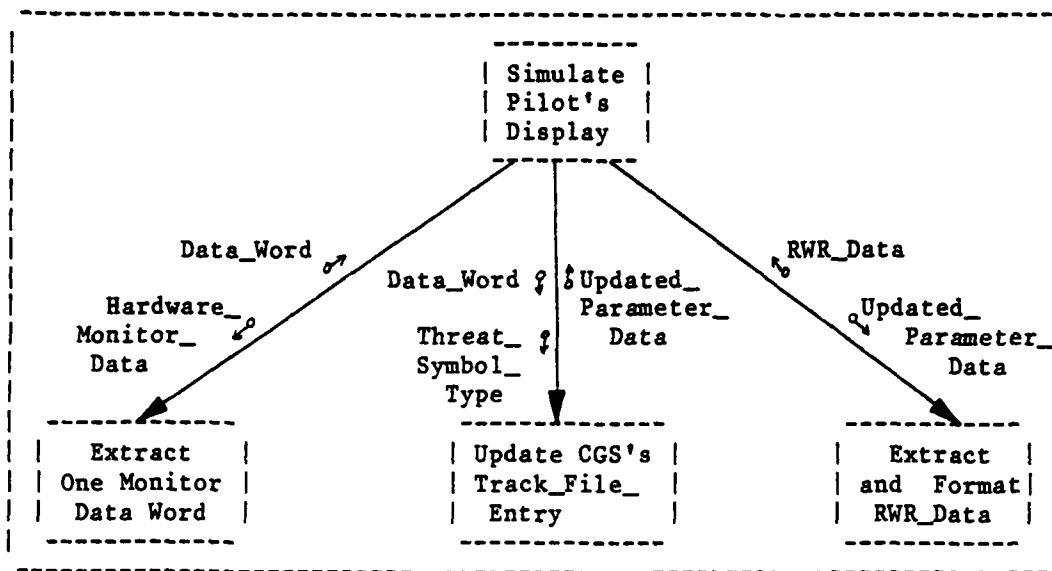


Figure 29. Simulate Pilot's Display Module Structure Chart

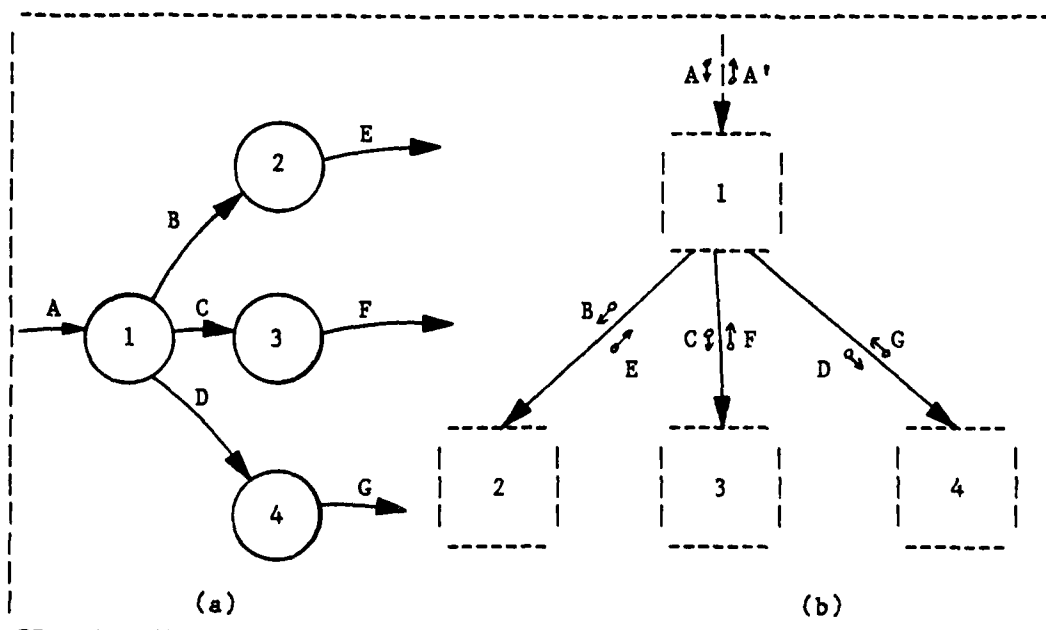


Figure 30. Transaction Analysis Technique

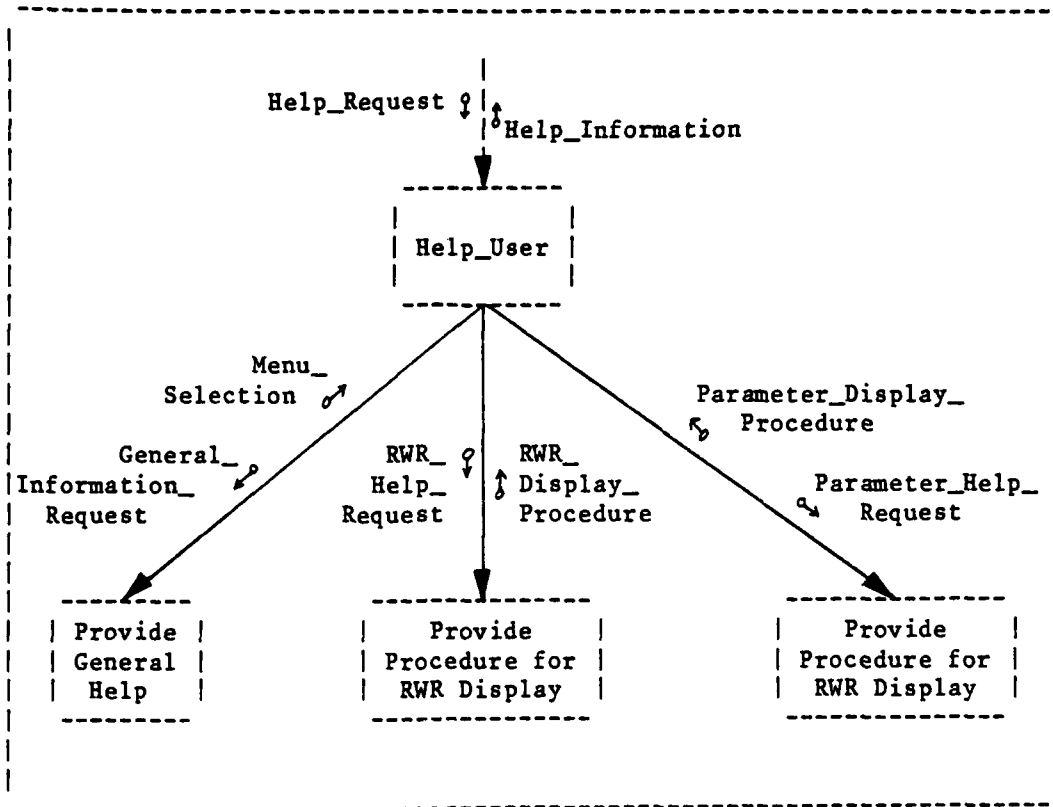


Figure 31. Help User Module Structure Chart

in Yourdon and Constantine's book, Structured Design (Ref 17:171-201). This technique was used throughout the module structure design phase. An example of this technique being used in the CGS design is shown in Figure 29 for the simulation of the pilot's display.

The other technique used a great deal was transaction analysis (Ref 17:202-222). This technique is particularly useful for translating DFD's with processes that classify an input as one of a number of outputs. Using transaction analysis, a DFD like that shown in Figure 30(a) may be translated into a module structure chart like that in

Figure 30(b). The module structure chart for the help-user command is shown in Figure 31 as an example of this technique.

Summary

This chapter translated the software requirements into logical and physical models of CGS. The logical model used data flow diagrams to emphasize the flow of data through CGS, while de-emphasizing procedural aspects of the problem and physical solutions. The physical model of CGS was derived primarily using transform and transaction analysis. These techniques are briefly described and the module structure charts are included in Appendix D. The module structure charts were used during the implementation of CGS, described in Chapter V.

V. Implementation and Testing of CGS

Introduction

The entire Computer Graphics System has been implemented. The unique factors which impacted this implementation are discussed in this chapter, as are the basic principles used in implementing the modules. Also, the test plan and procedures employed to verify the CGS modules and validate the system are described. Finally, the testing results are stated.

Implementation

One of the first challenges in the implementation phase was to select a system implementation strategy.

Strategy Two implementation strategies were considered, Top-down and Bottom-up. The Top-down strategy was chosen to implement CGS due to several advantages it offered. However, before these advantages can be understood, it is necessary to understand the basic concepts in each strategy.

Top-down design is a modular design strategy that creates a system design in terms of major functions, which are decomposed into more detailed functions. The top-down implementation suggests that high-level modules should be coded and tested first, followed by the next lower-level until all of the modules have been implemented

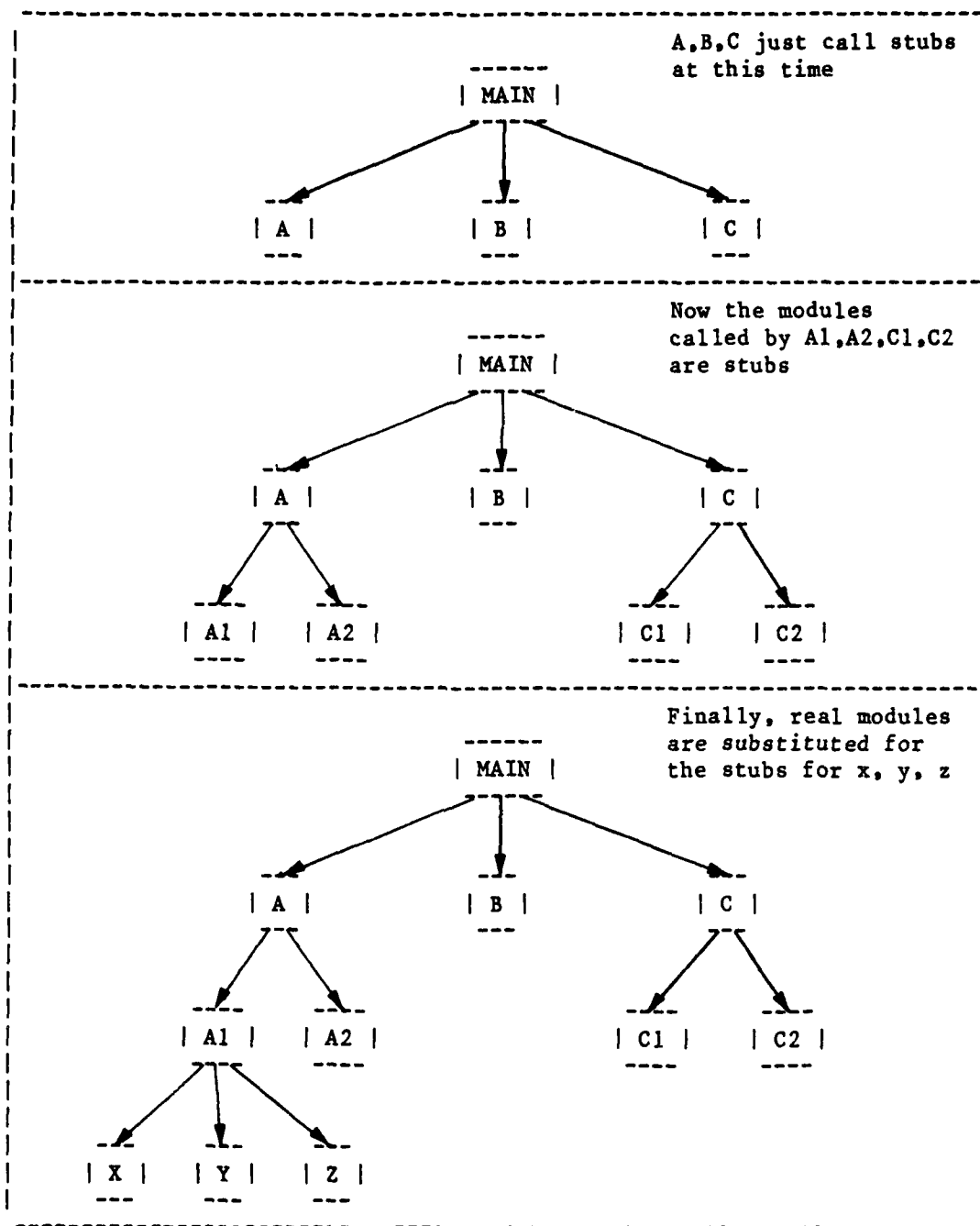


Figure 32. Top-Down Implementation

(Ref figure 32). High-level modules are tested using stubs to simulate the work of lower-level modules (Ref 8:216). Bottom-up design creates a system design in terms of minor functions, which are integrated together to form major functions. The bottom-up implementation suggests low-level modules should be coded and tested first, followed by the next higher-level until all of the modules have been implemented (Ref figure 33). The lower-level modules are tested using driver programs and specialized test data.

With these very abbreviated description of top-down and bottom-up strategies, it is now possible to examine the seven advantages of top-down implementation (Ref 17:340-358).

1. Unit, integration, and systems testing are all eliminated as separate phases. In effect, every time a new module (or small group of modules) is added to the system, an integration test is run. The addition of the last module in actuality represents the final system test.
2. Major interfaces in the system are tested early with a top-down, incremental testing strategy. Hierarchical modular designs identify high-level modular functions which can be coded and tested before detailed specifications have been developed for lower-level, detailed functions. As a result, high-level program functions and inter-program system's functions can be tested early, minimizing the likelihood of interface problems necessitating revision of lower-level modules.

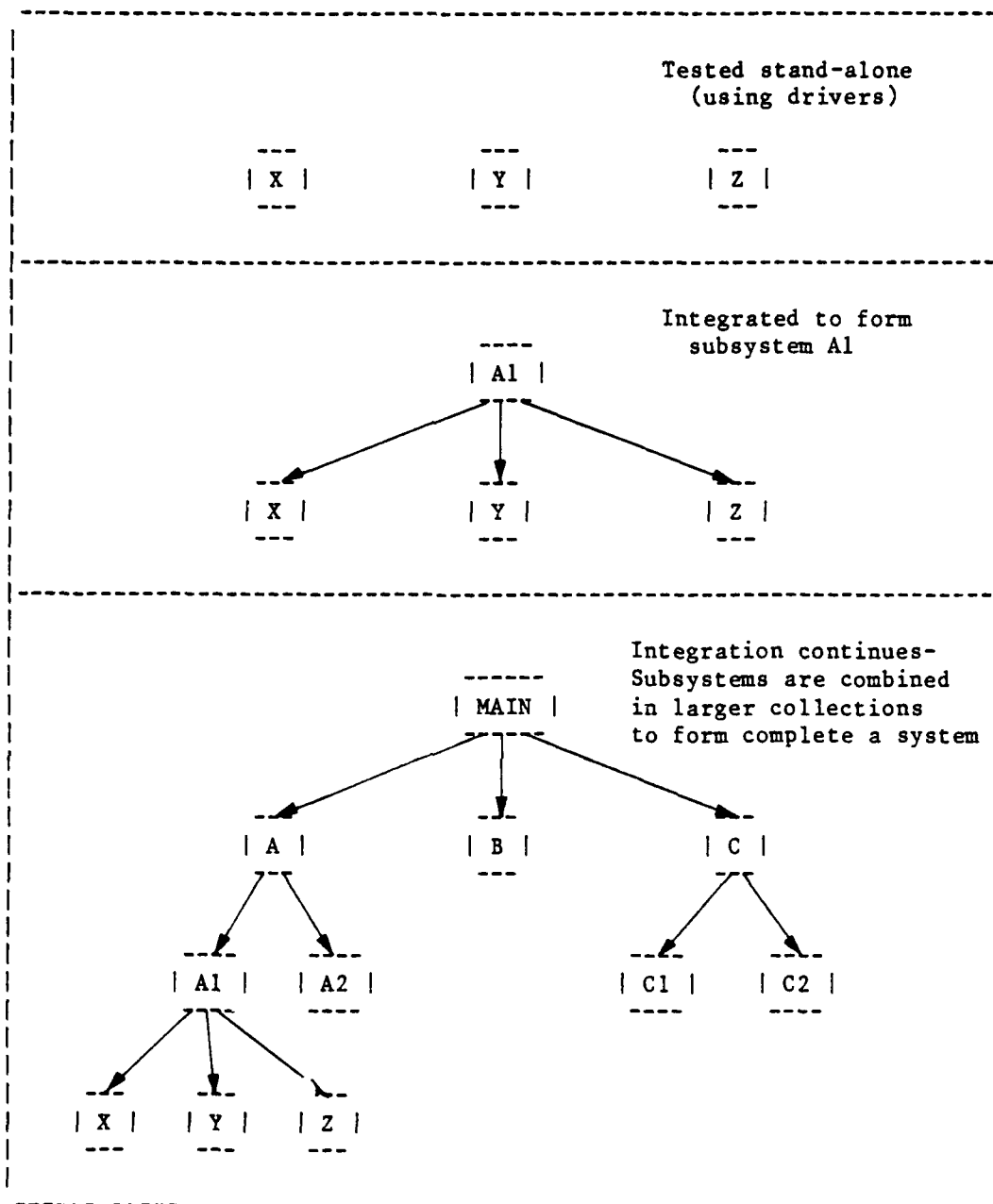


Figure 33. Bottom-Up Implementation

3. Users can see a working demonstration of a skeleton version of the system long before the entire system has been completed, thereby enhancing users' support for and involvement with the system.
4. Deadline problems are more manageable. Serious design flaws are exposed early in testing.
5. Debugging is easier. In a bottom-up, phased-testing environment, modules that have been successfully unit-tested are thrown together for an integration test. If the test fails, the bug could be anywhere in the system, causing considerable difficulty in debugging. With incremental testing in a top-down environment, a module or small group of modules is added to a working skeleton of the system. If the test fails in this case, debugging probably would be reasonably simple, because the bug would exist either in the newly added code or in the interface between the new code and the already tested, working skeleton of the system.
6. The need for test harnesses is eliminated. In a bottom-up environment, testing low level modules requires driver programs and specialized test data; the actual logic of high-level modules, in a top-down environment, drives lower-level modules. Module stubs simulate the work of even lower-level modules. Thus, no driver program or specialized test data is created only to be discarded later.
7. Requirements for machine time are distributed more evenly and are more predictable. In bottom-up implementation, testing begins

perhaps halfway through a project and grows slowly as modules are compiled and unit-tested. In top-down implementation, testing begins sooner than in the bottom-up approach. The amount of daily test-time increases rapidly early in a project and then increases gradually as modules are added incrementally to the system.

With the implementation strategy defined, the next step was to transform the structure charts described in Chapter IV into a format easier to implement. The Warnier-Orr diagram technique was chosen because it is simple, readable, and easy to implement. It proceeds with one step leading directly and logically to the next step.

The process stops when the last program hierarchical level defines one specific function or when the last level of structure defines a unique element which cannot be further divided. These criteria are readily identifiable.

Readability is of utmost importance to every activity following the design phase of software development. The design must be readable and understandable to those whose jobs will be to code, test and maintain the software.

The last advantage and one of the most important, practically, is the ease with which a Warnier-Orr design can be implemented. The format of the Warnier-Orr diagram graphically represents the three constructs which are necessary and sufficient for any program implementation : sequence, decision and iteration (looping).

AD-A119 252

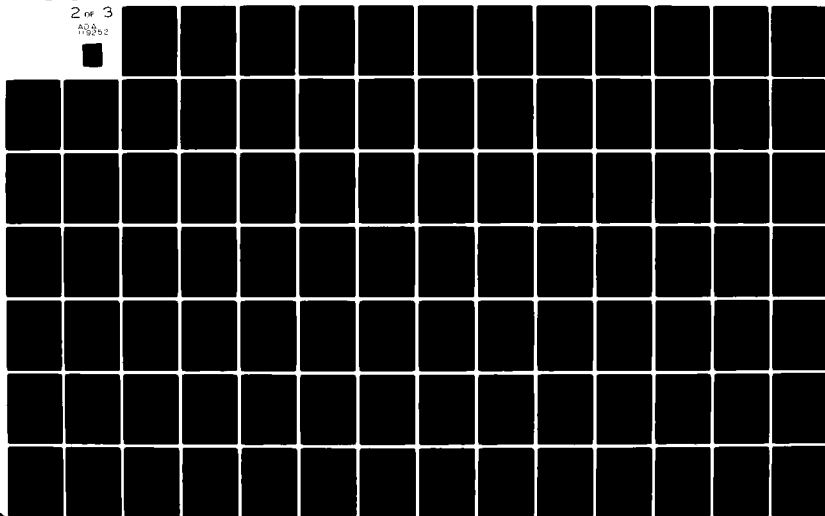
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCH00--ETC F/6 9/2
ALR-46 COMPUTER GRAPHICS SYSTEM FOR THE ROBINS AFB ELECTRONIC W--ETC(U)
DEC 81 J W THAMES
AFIT/6CS/EE/81D-18

UNCLASSIFIED

NL

2 of 3

AD-A119 252



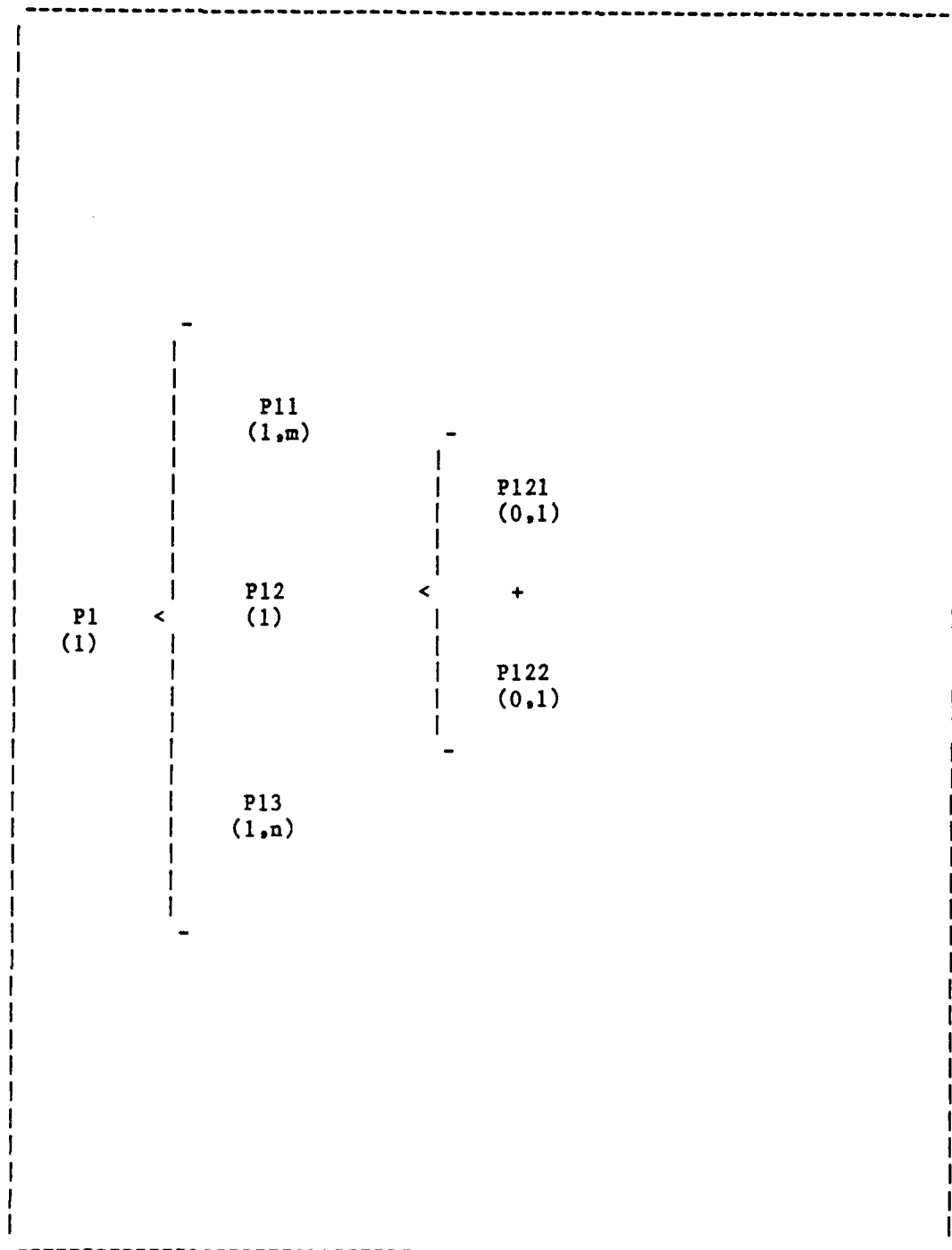


Figure 34. Warnier-Orr Process Structure Diagram

Table 6. List of all Warnier-Orr Diagrams

ALR46CGS

OPERATOR_INTERFACE
HELP

PARAMETER

CREATE_TRACK_FILE_ENTRIES
INITIALIZE_TRACK_FILE_ENTRY
READ_SYMBOL_TYPE_FILE
SET_STATE_OF_VT100
POSITION_CURSOR
GENERATE_FORM
UPDATE_TRACK_FILE_ENTRY
PROCESS_ONE_WORD
SHIFT
ALR46_DEGREES
OUTPUT_CRITICAL_PARAMETERS
POSTION_CURSOR

RWR_DISPLAY

CREATE_TRACK_FILE_ENTRIES
INITIALIZE_TRACK_FILE_ENTRY
READ_SYMBOL_TYPE_FILE
INITIALIZE_POSITION
SET_STATE_OF_4027
UPDATE_TRACK_FILE_ENTRY
PROCESS_ONE_WORD
SHIFT
ALR46_DEGREES
OUTPUT_TO_RWR_DISPLAY
DRIVE_PILOTS_DISPLAY
OUTPUT_A_SYMBOL
SEND_CRITICAL_PARAMETERS
GET_NEXT_COLOR
X_COORDINATE
Y_COORDINATE

In Figure 34 module P1 consists of three sequentially (top to bottom) executed steps (sequence) : P11, P12 and P13. Module P12 consists of two steps : P121 and P122. At least one of P121 and P122.

but not both, are executed each time P12 is executed (decision). The notations on P11 and P13 in parenthesis indicate how many times that module is repeated each time the parent module, P1, is executed. In the case of P11 and P13, the modules will be executed from one to "m" and one to "n" times, respectively. This represents looping. The notation on P1 indicates that the sequence P11-P12-P13 will be executed only once.

Extensions to the basic three constructs are obtained by combining them. A combination of the loop and decision constructs produces the DO-UNTIL or DO-WHILE constructs while multiple decisions illustrate the CASE statement of the PASCAL language. These extensions are mentioned to illustrate the relationships that the Warnier-Orr diagrams exhibit to implementation. All of the Warnier-Orr diagrams (Ref Table 6) developed for CGS is contained in Appendix E. An overview of CGS is shown in Figure 35.

Operator Interface. The first module to be implemented was a top-down design operator interface. The primary implementation goals were to make it easy to use and to provide the operator with help upon request.

The user has the option of entering either the full keywords for the commands or only the first letter of the keywords. This keeps a regular user of the system from being hampered by long commands, yet keeps the novice or infrequent user from having to cope with cryptic

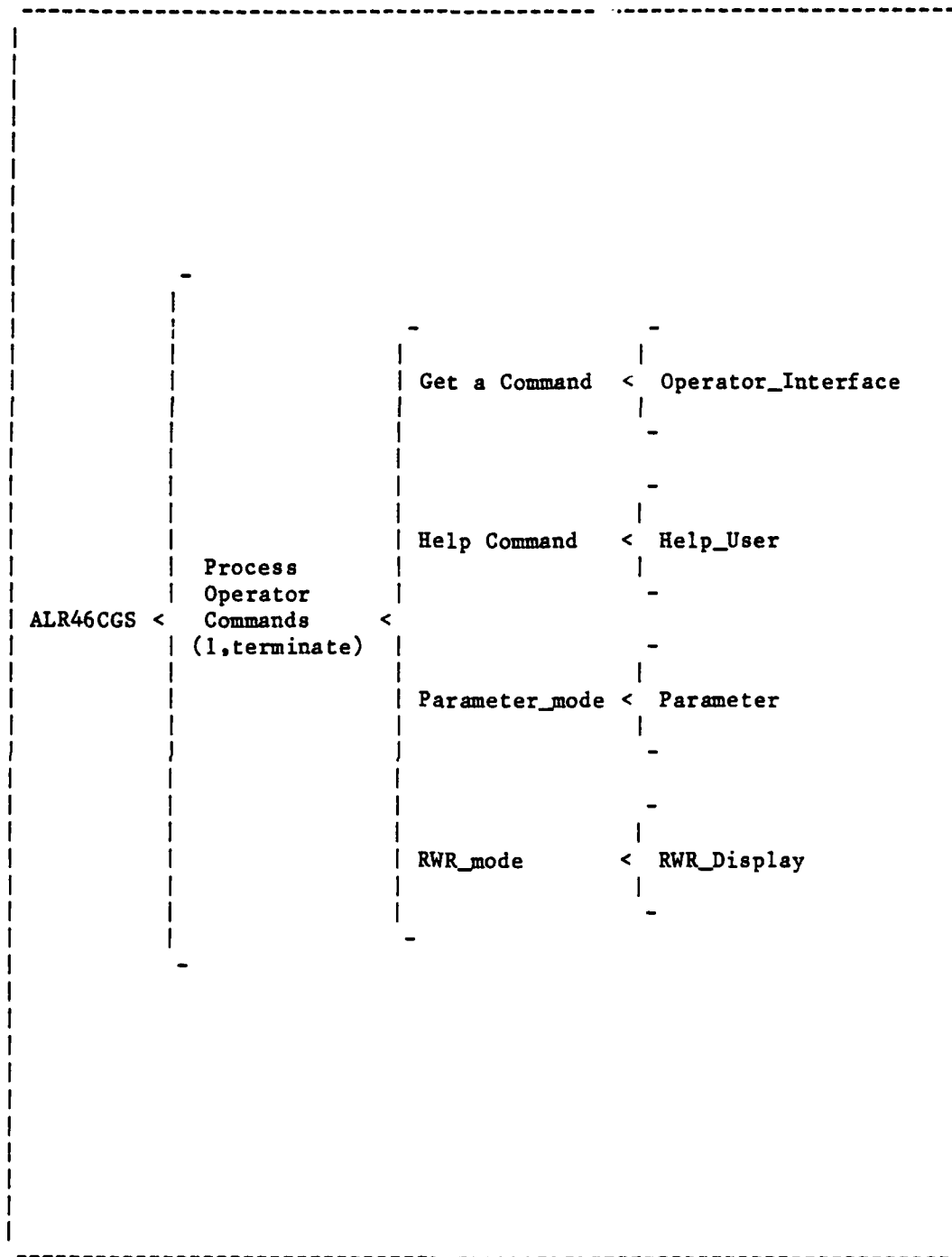


Figure 35. Warnier-Orr System Overview

Table 7. Valid Keywords

CGS Command Parameters

1. HELP
2. RWR DISPLAY
3. PARAMETER DISPLAY

one-letter commands and parameters. The list of keywords in the CGS command language is given in Table 7.

The user "help" protocol was implemented and can provide the user all information required to use CGS. This module is tiered, so entry into the procedure can be accomplished by simply typing "help".

Data Structures Careful consideration was given to data structures that are to be employed. For example, the tradeoffs of using a linked list verses an array to store the information needed to drive the display was thoroughly studied before the linked list was chosen. The linked list offered fast access to each critical parameter and was the same structure used in the ALR-46 OFF. The linked list also allowed the number of track file entries to vary with the number of active threats and would on the average require less storage space than the array. On the other hand, the array offered direct access to key parameters and was simpler to implement. Even though the linked list was more difficult to implement, it was easier for the ALR-46 personnel to maintain because it was the same as the structure used in the OFF.

The requirement for ease of maintenance made the linked list more attractive.

Parameter / RWR Processes During implementation, every attempt was made to continue the practices employed during the requirements and design. As shown in Figure 36 and 37, both processes were implemented in a modular, top-down manner. Also each module was implemented such that it hid the data structures and algorithms employed as much as possible from other modules. One example of this was in the RWR symbol display module. This module was passed the symbol and its location on the display. Thus, the subordinate modules did not require any knowledge of the actual threat the symbol represented. The last phase of implementation for each module consisted of debugging the module. Care was taken to insure changes made in this phase did not destroy the structure of the modules originally implemented. The code and Warnier-Orr diagrams for these modules are included in Appendix E. Once the modules were free of compilation errors, they were tested using stubs and drivers, as well as the techniques described in the following section.

Testing

A top-down approach was used to test CGS. Each module was tested before another module was integrated into CGS. Therefore, any problems encountered were immediately isolated to the new module. Two testing strategies were employed, "black-box" and "white-box". Black-box testing requires all test data to be derived solely from the

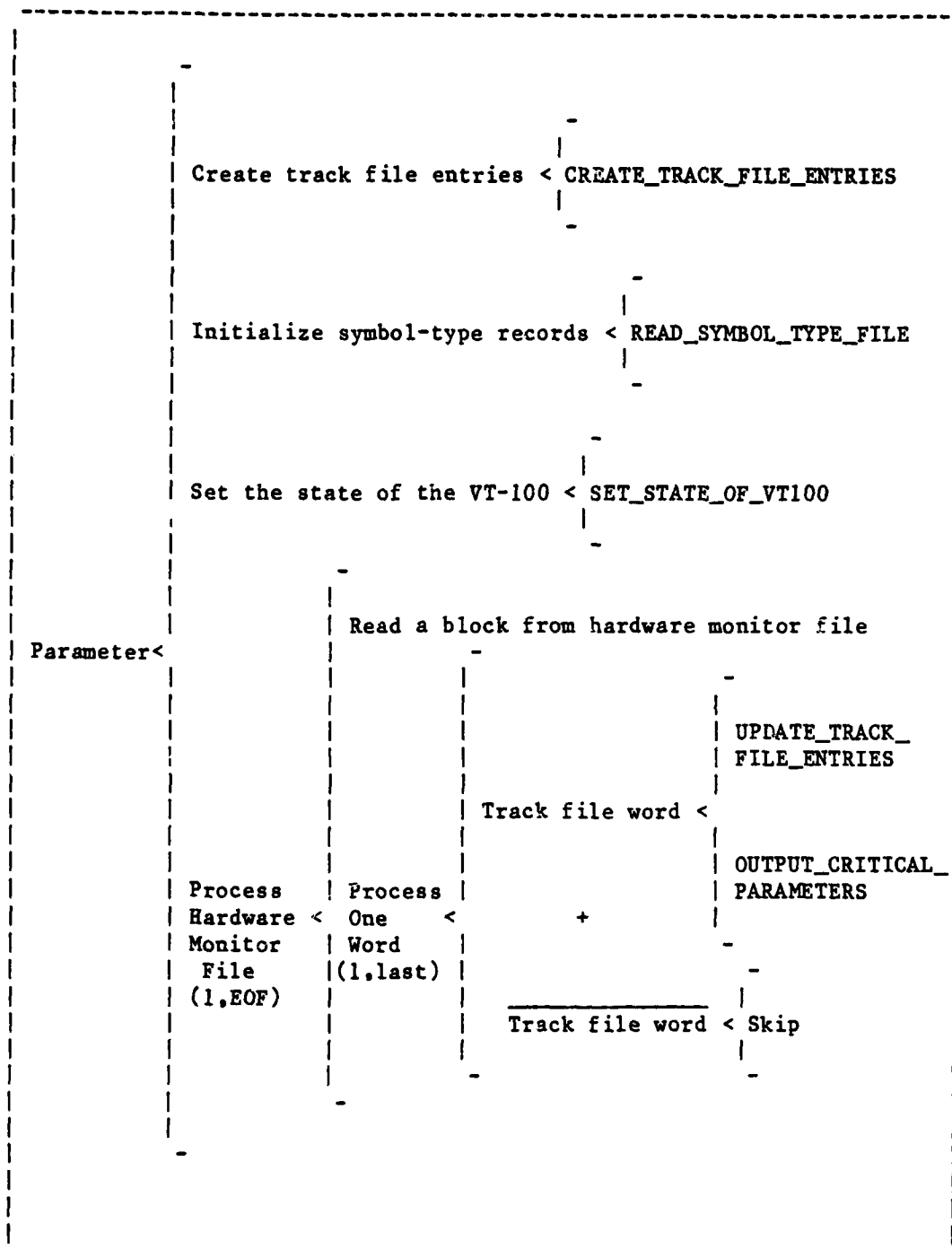


Figure 36. Warnier-Orr Diagram of Parameter Process

specifications, while white-box testing allows test data to be derived from an examination of the program's logic (and often, unfortunately, at the neglect of the specification)(Ref. 18:9).

White-box testing was used to test each module as it was integrated into CGS. Inputs were chosen to insure that all segments of code were executed. Also, when a branch depended upon a compound logical expression, inputs were chosen for each of the possible combinations.

Equivalence partitioning, a form of black-box testing, was also used. This procedure entailed partitioning the input domain of CGS into a finite number of equivalence classes so one could reasonably assume a test of a representative value from each class was equivalent to a test of any other value. For example, the azimuth parameter has a legal range from zero to two hundred and fifty nine. The legal values were divided into sixteen equal subranges starting with zero. Since any azimuth value from a specific subrange was the same as any other value from that subrange, only four values were needed to perform this type test.

Boundary value analysis, another form of block-box testing, was also used to test the modules. The values at the borders of the equivalence classes were tested to detect "off by one" errors. An example of this technique is the testing of the stagger level parameter. This parameter has a legal range of zero to seven. To test the behavior of the module that decoded the stagger level, stagger level values of zero, one, seven, and eight were used.

The testing was done incrementally. As a module was coded and debugged, it was then integrated into the set of modules that had already been tested. This new set of modules was then rigorously tested using the techniques already mentioned. This procedure was repeated until the test of the last module was actually a test of the complete CGS. This method worked well and allowed interface problems between modules to be isolated quickly. Finally, records were kept of these tests to aid in maintaining the program. Using these documented results, a modification to CGS can be verified using the same input data in addition to new data to test the extensions or modifications to CGS. These test results are included in Appendix G.

Summary

The Computer Graphics System was implemented on the VAX 11/780, but care was taken to ensure its compatibility with the PDP 11/34 by only using Pascal statements that were compatible with both the VAX 11/780 and the PDP 11/34. The operator interface was implemented to accommodate both the novice and expert user. The RWR and Parameter modes were implemented as defined in the system specification. Both information hiding and a careful selection of data structures were employed during the implementation. Finally, the modules were tested incrementally using path analysis, equivalence class testing, and boundary value analysis.

VI. Conclusions and Recommendations

This investigation, the development of the CGS, was based on the requirements provided by the Electronic Warfare Engineering Branch Laboratory. User interviews were used to determine the user requirements.

User requirements and the system constraints were used to develop the CGS system requirements. This portion of the investigation was very time consuming since design decisions had to be made. Next, the software requirements were defined in more detail.

The time invested in the system requirements proved worthwhile throughout the investigation. The design proved to be straight forward because of the amount of partitioning that had already been done. Structured design techniques were employed in the development of both the logical and physical models of CGS.

The software modules were implemented using structured techniques. The most difficulty was encountered in controlling the DEC VT-100 CRT and the Tektronix 4027 CRT. Many of the specialized commands used had limited documentation. Each command was documented in CGS at the time it was used.

The testing was conducted using branch analysis, equivalence class testing, and boundary value analysis. The CGS modules were tested incrementally which greatly simplified the overall system testing.

In summary, through the user interviews, Structured Analysis techniques, Structured Design techniques, and top-down design techniques, CGS went from an idea to a fully operational system. All of the requirements of the EW Engineering Branch Laboratory were satisfied. Tests performed with CGS on AFIT's VAX 11/780 using simulated data indicated CGS will be a valuable test tool for the ALR-46 personnel. The sponsor felt that CGS did meet the objective of the investigation, which was to reduce the time required to modify the ALR-46 system. CGS will be installed on the sponsor's computer in 1982.

Recommendations

Because of the structured approach used to develop CGS, enhancements and additional capabilities can be added without affecting the current capabilities. It is recommended the following tasks be considered by follow-on investigations in order to realize the full potential of CGS:

1. Provide a capability to determine the amount of time used by each procedure in the ALR-46 Operational Flight Program. As the threat environment becomes more dense, the number of threats that must be simultaneously processed by the ALR-46 increases. The time required to process a threat must be reduced if the system is to continue meeting its response time requirements. This CGS capability would allow the large users of CPU time to be identified and thus the prime candidates for optimization would also be identified.

2. The response time of CGS could be reduced significantly if DEC's "QIO" system procedure was used for I/O instead of the standard read and write statements. Tests on the PDP 11/34 indicated a ten to one difference in speed between a standard read/write and QIO (Ref 11).
3. The CGS Output_A_Symbol procedure should be enhanced to include additional threat symbols.
4. The operator interface procedure should be transformed into a standalone procedure that sends and receives messages to and from the remainder of the CGS procedures. This would allow the RWR and Parameter modes to operate simultaneously and also give the operator control over CGS at any point in time. At the present time, the operator has to wait for one mode to complete before another mode can be selected.

Bibliography

1. Alden, A.L., et al. Electronic Countermeasures. Los Altos: Peninsula Publishing.. 1978.
2. Digital Equipment Corporation. VAX-11 Pascal Language Reference Manual. Description of Pascal Language. Software Distribution Corporation, Maynard, Massachusetts 01754, November 1979.
3. Digital Equipment Corporation. VAX-11 Pascal User's Guide. Description / Examples of the Pascal Language. Software Distribution Corporation, Maynard, Massachusetts 01754, November 1979.
4. Levy, Henry M. and Richard H. Eckhouse Jr. Computer Programming and Architecture The VAX-11. Bedford: Digital Press, 1980.
5. Tektronix. 4027 Color Graphics Terminal Operator's Manual. Introduction to the Operation of the 4027 CRT. Tektronix, INC., PO. Box 500, Beaverton, Oregon 97077, January 1980.
6. Tektronix. 4027 Color Graphics Terminal Programmer's Reference Manual. Description of the 4027 Command Language. Tektronix, INC., PO. Box 500, Beaverton, Oregon 97077, October 1979.
7. EK-VT100-UG-002. VT100 User's Guide. Operator's / Programmer's User's Guide. Maynard, MA: Digital Equipment Corporation, 1979.
8. Weinberg, Victor. Structured Analysis. New York : Yourdon Press, 1979.
9. Digital Equipment Corporation. PDP-11 Processor Handbook. Description of PDP-11/04/34/44/60/70. Digital Sales Support Literature Group, 1979.
10. Digital Equipment Corporation. VAX Hardware Handbook. Introduction to VAX-11 Hardware Elements. Digital Sales Support Literature Group, 1980.
11. Digital Equipment Corporation. VAX Software Handbook. Description of the VAX operating System. Digital Sales Support Literature Group, 1980.
12. Hayes, Phil et al. "Breaking the Man-machine Communication Barrier". Computer. 3:19-29 (March 1981).

Bibliography (cont.)

13. Comtek. "AN/ALR-46 Integration Support System User Guide". Describes how to use the ALR-46 test station. One Technology Center, 45 Oak Street, Buffalo, New York 14203, August 1981.
14. Newman, William M. and Robert F. Spoull. Principles of Interactive Computer Graphics (second edition). New York : McGraw-hill Book Company, 1979.
15. Tesler, Larry. "The Smalltalk Environment". Computer. 3:90-147 (August 1981).
16. Softech, Inc. "An Introduction to SADT--Structured Analysis and Design Technique". Technical Report 9022-78. Softech, Inc. Waltham, Massachusetts. November, 1976.
17. Yourdon, Edward and Larry L. Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and System Design. Englewood Cliffs, N.J.: Prentice Hall, Inc., 1978.
18. Myers, Glenford J. The Art of Software Testing. New York : John Wiley and Sons, Inc., 1979.
19. Higgins, David A., "Structured Programming with Warnier-Orr Diagrams". Byte, 1978.
20. WRALC. "Software Change Processing/Configuration Management for EW Systems". Describes the procedure for modifying the ALR-46. MMROI 55-1. Robins AFB, GA 31093, August 1981.

APPENDIX A

THE ALR-46 EMITTER TRACK FILE

This Appendix contains a description of the ALR-46 emitter track file entries used by CGS to generate the Parameter and RWR displays. Actually, the emitter track file is not a file, but a buffer within the ALR-46 Operational Flight Program. The track file is divided into sixteen records which contain sixteen words each. All records have identical formats. The structure of the emitter track file is shown in Figure 38. Only the bits used by CGS are described. The function of most of the bits not described is classified. Word zero is used by the ALR-46 to form the linked list. This structure is discussed in the description forward zero which follows.

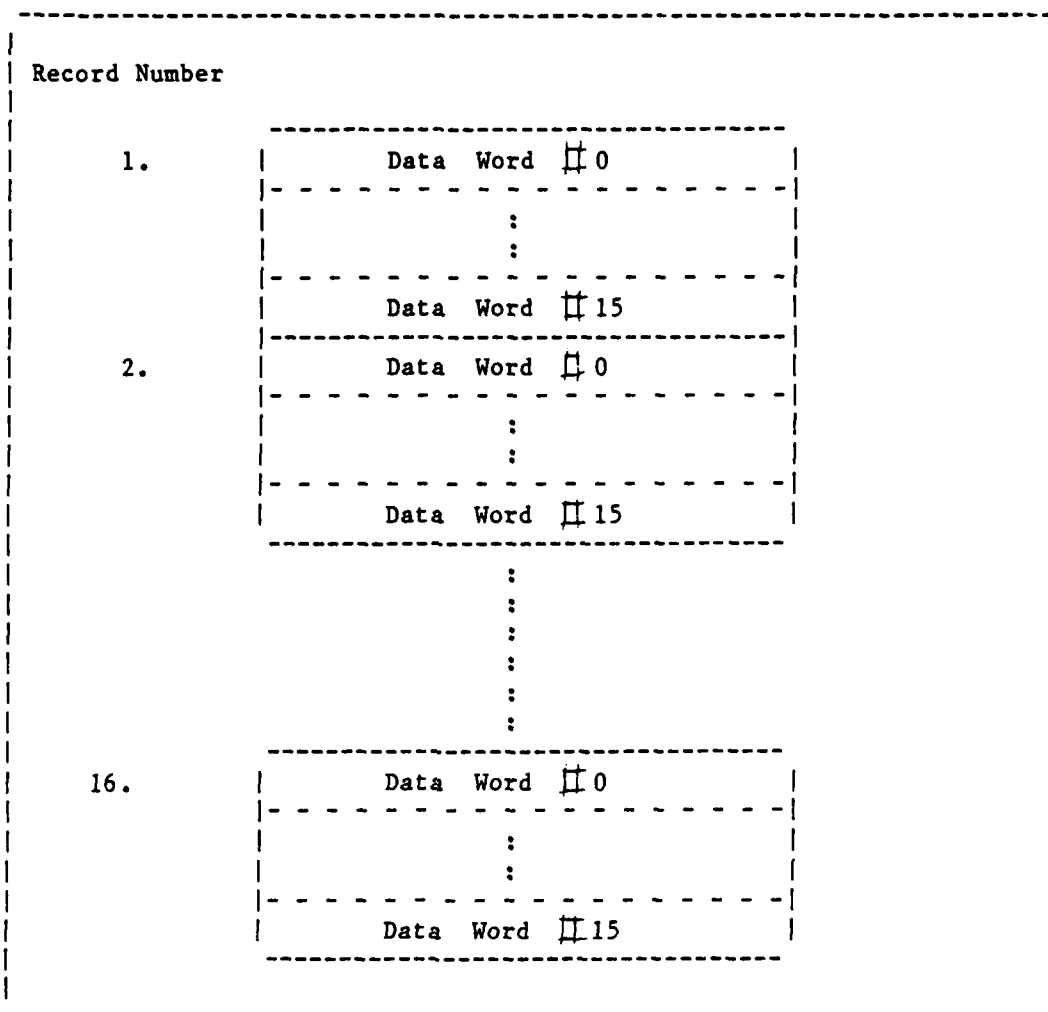


Figure 38. Record Structure of the ALR-46 Emitter Track File

The following is a description of the emitter track file words used by CGS: .

WORD 0:

BITS 0 - 15

1. Bits 0 - 15: A sixteen bit address is contained in bits 0-15.

To understand the function of this address, the reader must understand how the emitter track file is accessed and maintained. The OFP, which resides in the memory of the ALR-46 Flight Processor, maintains a set of pointers (first_available_record and first_emitter) which point to records in the track file (Ref Fig 17). If the first_available_record points to itself, no records are available. If the first_emitter pointer points to itself, no emitters are active. As shown in Figure 17, the records in the track file form two linked lists, the "Red_Link_List" and the "Blue_Link_List". The Red_Link_List is composed of records of active emitters, while the Blue_Link_List contains all of the records available for use. The first_emitter_pointer is the beginning of the Red_Link_List. It points to the highest priority threat, the highest priority threat points to the next highest priority threat, etc., until the lowest priority threat is reached. The lowest priority threat points to the first_emitter_pointer. The first_available_emitter pointer is the beginning of the

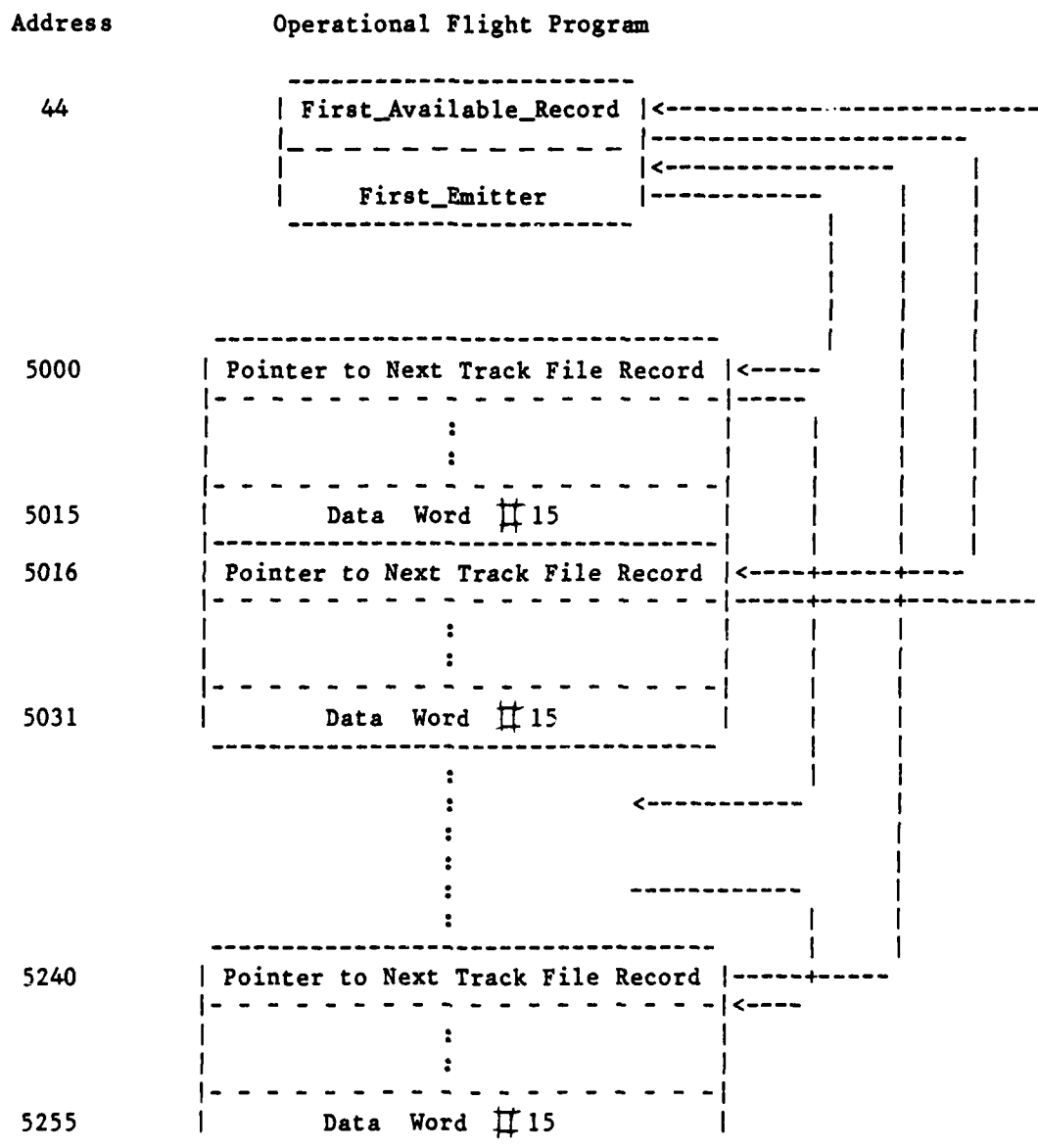


Figure 17. ALR-46 Emitter Track File

note: The pointer arrangement shown depicts fifteen used records (active emitters) and one available record.

Blue_Link_List. It points to the first available record, the first available record points to the next available record, etc., until the last available record is reached. The last available record points to the first_available_emitter pointer. When an enemy threat is recognized by the ALR-46, it is assigned a record in the track file, based on the threat's priority. The pointers have to be updated in both linked lists, since an available record has to be removed from the "Blue_Link_List" and inserted into the "Red_Link_List". As the ALR-46 system collects more information about a threat (PRI, PW, etc.), the OFP updates the appropriate word(s) in the record associated with that threat. Word 0 is the key to the structure of the emitter track file.

WORD 2:

BITS 0 - 15

1. Bits 0-15: The sixteen bit address contained in bits 0-15 is used to define the symbol to be displayed. The address is an index into the symbol / type table. Refer to Appendix B for additional information.

WORD 3:

BITS 0 - 15

1. Bits 0-15: Either the frame PRI or the average PRI is contained in bits 0-15. If the pulse train is staggered, word 3 contains the frame PRI, otherwise it contains the average PRI. The units of time for word 3 is tenths of microseconds.

WORD 4:

BITS 8 - 15

1. Bits 8-15: The Hi Band Sine is contained in bits 8-15. It is used by the ALR-46 along with hi band cosine to determine the threat's azimuth. The hi band sine can range from zero to two hundred and fifty six.

WORD 5:

BITS 8 - 15

1. Bits 8-15: The hi band cosine is contained in bits 8-15. It is used by the ALR-46 along with hi band sine to determine the threat's azimuth. The hi band cosine can range from zero to two hundred and fifty six.

WORD 6:

|BITS 10-15 |BITS 5-9 |BIT 3|BIT 2|BIT 1|BIT 0|

1. Bit 0: If bit 0 is set (equal to one), a flashing circle is put around the symbol when it is displayed.
2. Bit 1: If bit 1 is set, a steady circle is put around the symbol when it is displayed. If both bit 0 and bit 1 are set, then bit 0 governs.
3. Bit 2: If bit 2 is set, a diamond is placed around the symbol when it is displayed.
4. Bits 5-9: The priority is contained in bits 5-9. The priority has a range from zero to thirty-two.
5. Bits 10-15: The Priority / Round is contained in bits 10 - 15. The priority / round has a range from zero to fifty six.

WORD 7:

| |Bit 9| |Bit 4|Bit 3|Bit 2|Bit 1|Bit 0|

1. Bit 0: If bit 0 is set, the ALR-46 has identified a threat whose frequency is in band zero.
2. Bit 1: If bit 1 is set, the ALR-46 has identified a threat whose frequency is in band one.

3. Bit 2: If bit 2 is set, the ALR-46 has identified a threat whose frequency is in band two.
4. Bit 3: If bit 3 is set, the ALR-46 has identified a threat whose frequency is in band three.
5. Bit 4: If bit 4 is set, the ALR-46 has identified a threat whose frequency is in band four.
6. Bit 9: If bit 9 is set, the pulse train is jittered.

WORD 8:

 | BITS 11-15 |BITS 8-10 |BITS 5-7| |BITS 0-2|

1. Bits 0-2: The Conditional Emitter Program Count (CEPC) is contained in bits 0-2. The CEPC has a range from zero to seven.
2. Bits 5-7: The ML Age is contained in bits 5-7. ML Age has a range of zero to seven.
3. Bits 8-10: The stagger level is stored in bits 8-10. The ALR-46 subtracts one from the stagger level count before storing it. Therefore, one must be added to stagger level before it is displayed. The ALR-46 subtracts one from the stagger level in order to store the maximum stagger level of eight into a three bit field. The stagger level has a display range of one to eight.

4. Bits 11-15: The low band age is contained in bits 11-15. The low band age has a range from zero to thirty-two.

WORD 10:

| BITS 6-12| BIT 5|

1. Bit 5: Bit 5 is the sign bit for the value contained in bits 6-12. If it is set, the value in bits 6-12 is a negative number in ones's complement form. Otherwise, the value in bits 6-12 is a positive number.
2. Bits 6-12: The Y_Displacement value is contained in bits 6-12. This value combined with the X_Displacement value is used by CGS to calculate the threat's range and azimuth. For additional information, refer to Appendix C.

WORD 12:

BITS 0 - 15

1. Bits 0-15: The mean PRI is contained in bits 0-15. Word 12 is the same as word 3, except for staggered pulse trains. When the pulse train is staggered, word 12 contains the Frame PRI. The unit of time for word 12 is tenths of microseconds.

WORD 13:

| BITS 6 - 15 | | BIT 1 | |

1. Bit 1: If bit one is set, Special is equal to YY. Otherwise, special is equal to NN.
2. Bits 6-15: The value of Deviation is contained in bits 6-15.

WORD 14:

| | BITS 3 - 7 | |

1. Bits 3-7: The High Band Age Count is contained in bits 3-7. It has a range from zero to thirty-two.

WORD 15:

| | Bits 0 - 4 | |

1. Bits 0-4: The record number is contained in bits zero through four. The record number has a range from zero to sixty-three.

APPENDIX B

THE SYMBOL / TYPE FILE

This Appendix contains a description of the CGS Symbol/Type file. It is used by CGS in conjunction with word two of the emitter track file to determine the threat's symbol and type. The file, which is part of CGS's input data, must exist prior to running CGS. The file contains sixty four sequential records, each three words in length. The actual data contained in the CGS Symbol/Type file is classified. With the approval of the sponsor, the author generated an unclassified file that could be used to develop CGS.

As shown in Figure 39, the symbol/type file contains two symbols and a type number for each entry. When the ALR-46 displays a threat, it alternates between symbol one and symbol two. Symbol one and two are always the same for threats the ALR-46 can uniquely identify. Therefore, the pilots only see one symbol. Unfortunately, certain threats have such similar characteristics, the ALR-46 can not distinguish between them. In this case, symbol one and symbol two will not not be the same. The pilot will actually see two different symbols alternately displayed at approximately once per second. The CGS symbol/type file contains entries for all of the threats the ALR-46 can uniquely identify, as well as entries for all pairs of threats that can

OFFSET	SYMBOL ONE	SYMBOL TWO	TYPE NUMBER
0	SA 1	SA 1	100
1	SA 2	SA 2	101
2	SA 3	SA 3	102
3	SA 4	SA 4	103
4	SA 5	SA 5	104
:	:	:	:
:	:	:	:
:	:	:	:
:	:	:	:
:	:	:	:
61	AI	-A-	161
62	2 BAR	SA 4	162
63	AAA	-2-	163

Figure 39. Symbol / Type File Structure

not be uniquely identified. Each entry in the file has a unique type number.

The address in word two of the emitter track file consists of the base address of the symbol table plus the offset into the symbol table. The base address of the symbol table must be subtracted from word two before it can be used as an offset into the symbol/type file.

APPENDIX C

CGS RANGE and AZIMUTH CALCULATION

This Appendix contains a description of the process used to calculate the threat's range and azimuth. Certain characteristics of the ALR-46 made the process of calculating a threat's range and azimuth very unique. These characteristics, shown in Figure 40, are listed below:

1. The ALR-46 coordinate system is shifted 45 degrees in the clockwise direction.
2. The X coordinate is positive in the left direction instead of the right.
3. The negative X and Y coordinates are in one's complement format.
4. The reference for zero degrees is shifted ninety degrees in the ALR-46 coordinate system as compared to the polar coordinate system.

The first step was to convert any negative X or Y coordinates to DEC's standard format for negative numbers. The range was easily calculated using

$$\text{Range} = \text{Square Root} (X ** 2 + Y ** 2)$$

The calculation the azimuth was more complicated than the range calculation. First, the azimuth was calculated using ALR-46 coordinate system, and it was then translated into the ALR-46 azimuth using

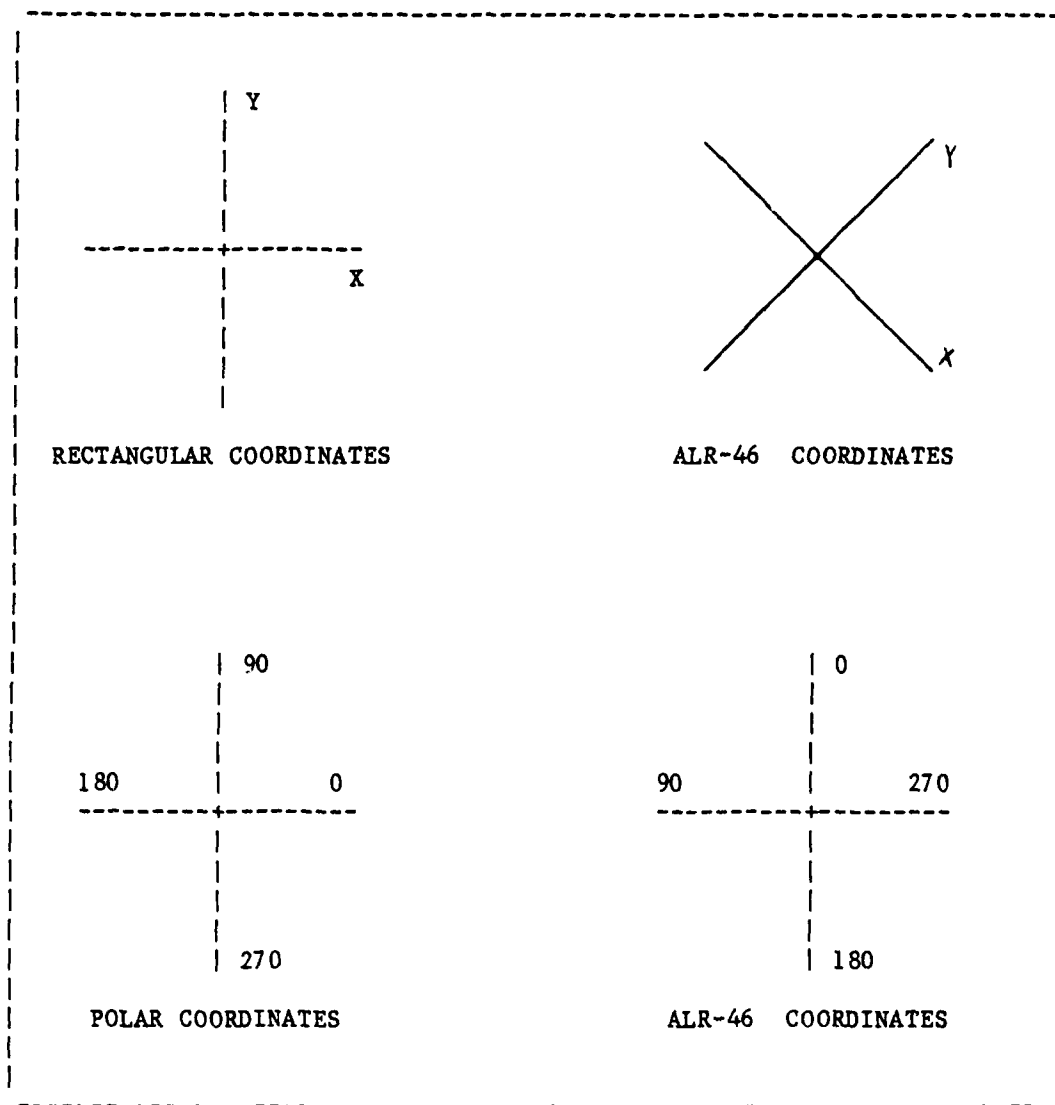


Figure 40. ALR-46 COORDINATE SYSTEM

the values of X and Y in the ALR-46 coordinate system. Refer to the "Process_One_Word" algorithm in the source listing for additional information.

APPENDIX D

MODULE STRUCTURE CHARTS

This Appendix contains the complete set of module structure charts needed to specify the design of the software for CGS. These charts were drawn using Yourdon and Constantine's structured design techniques (Ref 17).

List of Data Flow Diagrams	PAGE
System Structure Chart	115
Display Critical Track File Parameters	116
Update CGS's Track File Entry	117
Simulate Pilot's Display	118
Extract / Format RWR Data	119
Help User	120

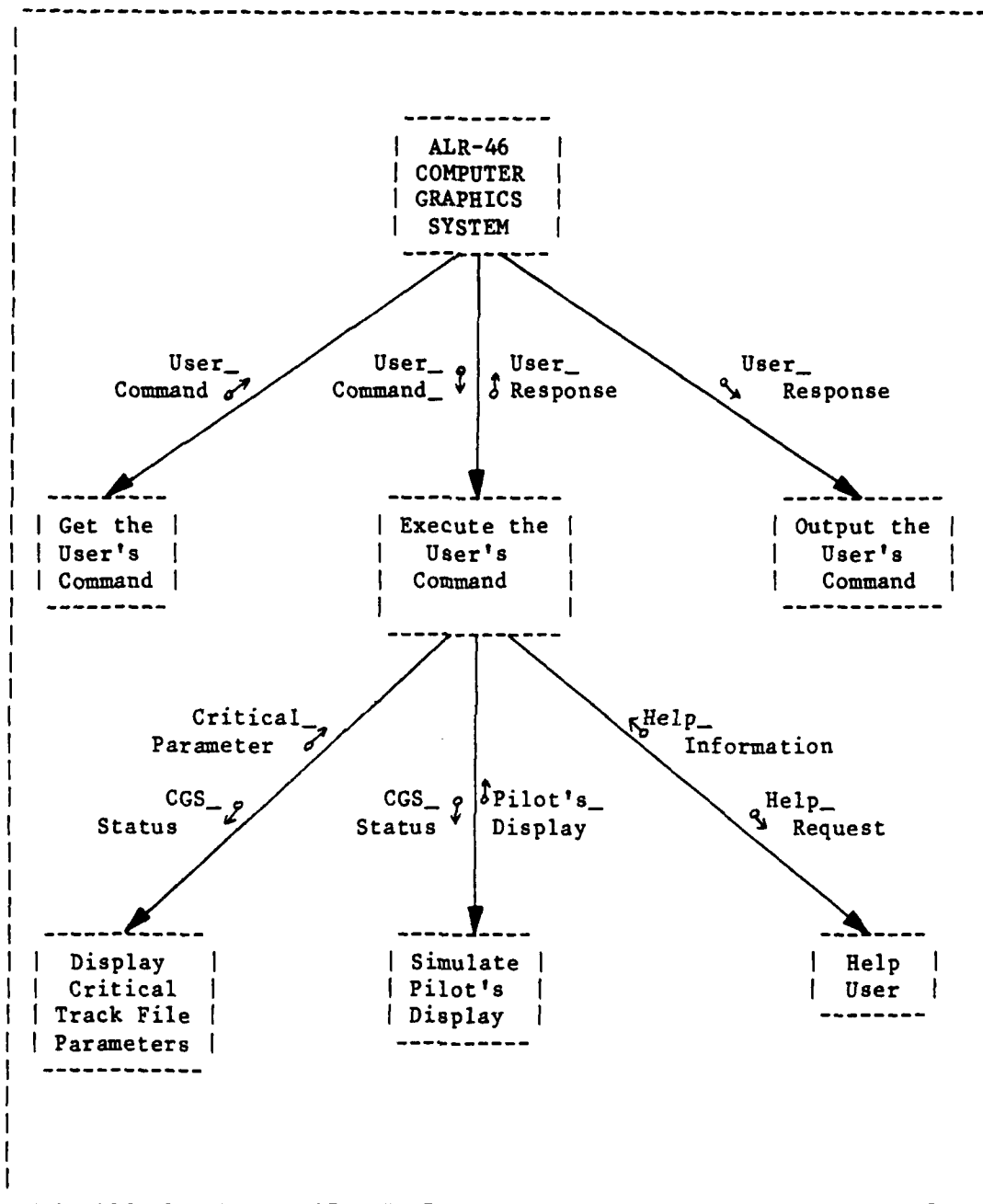


Figure 41. System Structure Chart

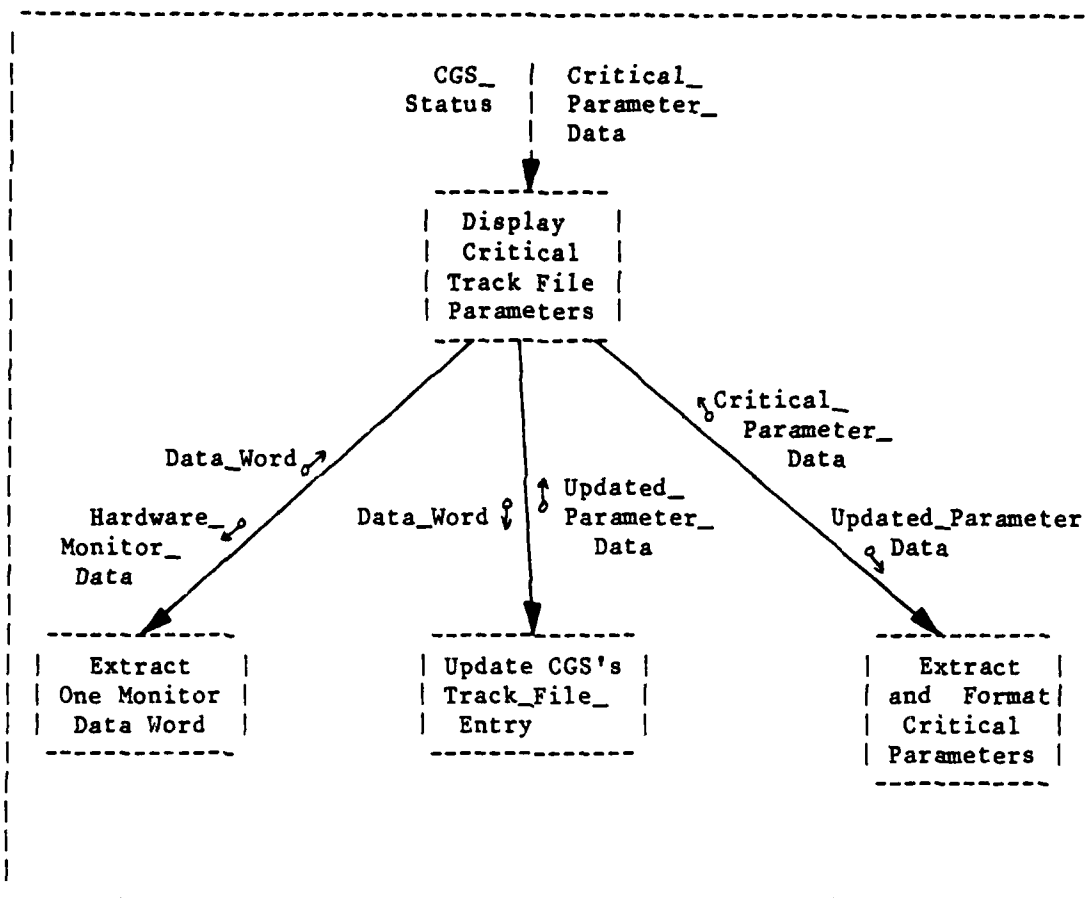


Figure 42. Display Critical Track File Parameters Structure Chart

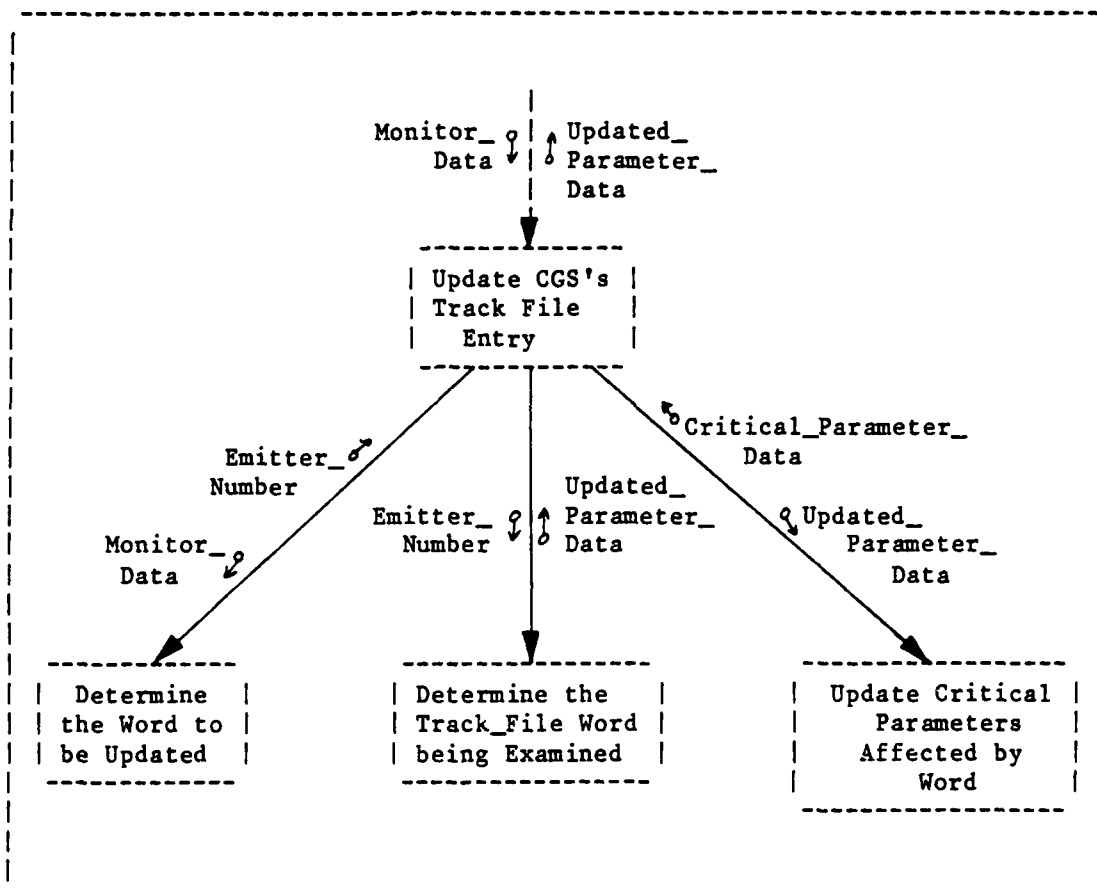


Figure 43. Update CGS's Track File Entry Structure Chart

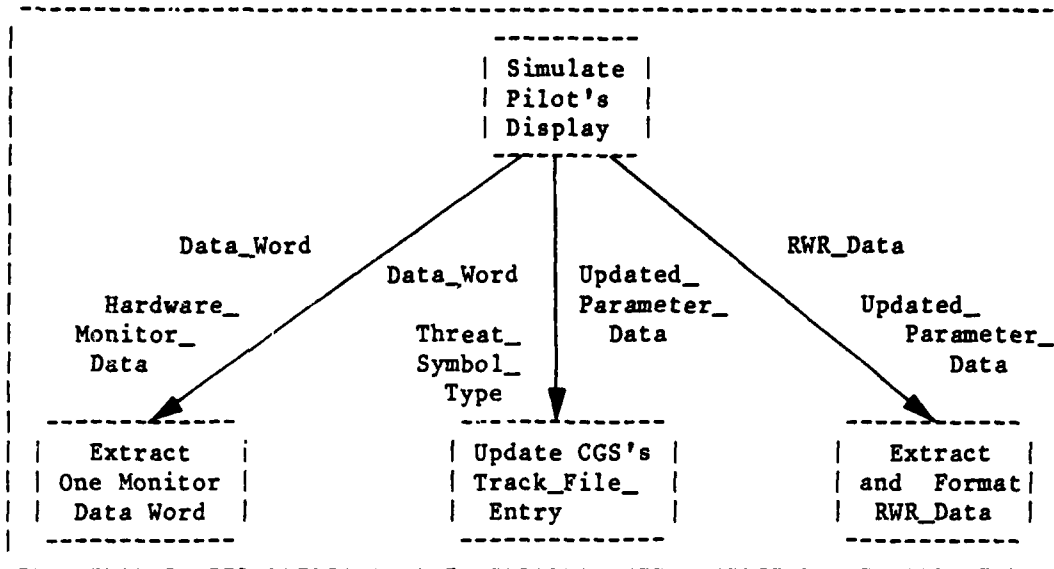


Figure 29. Simulate Pilot's Display Module Structure Chart

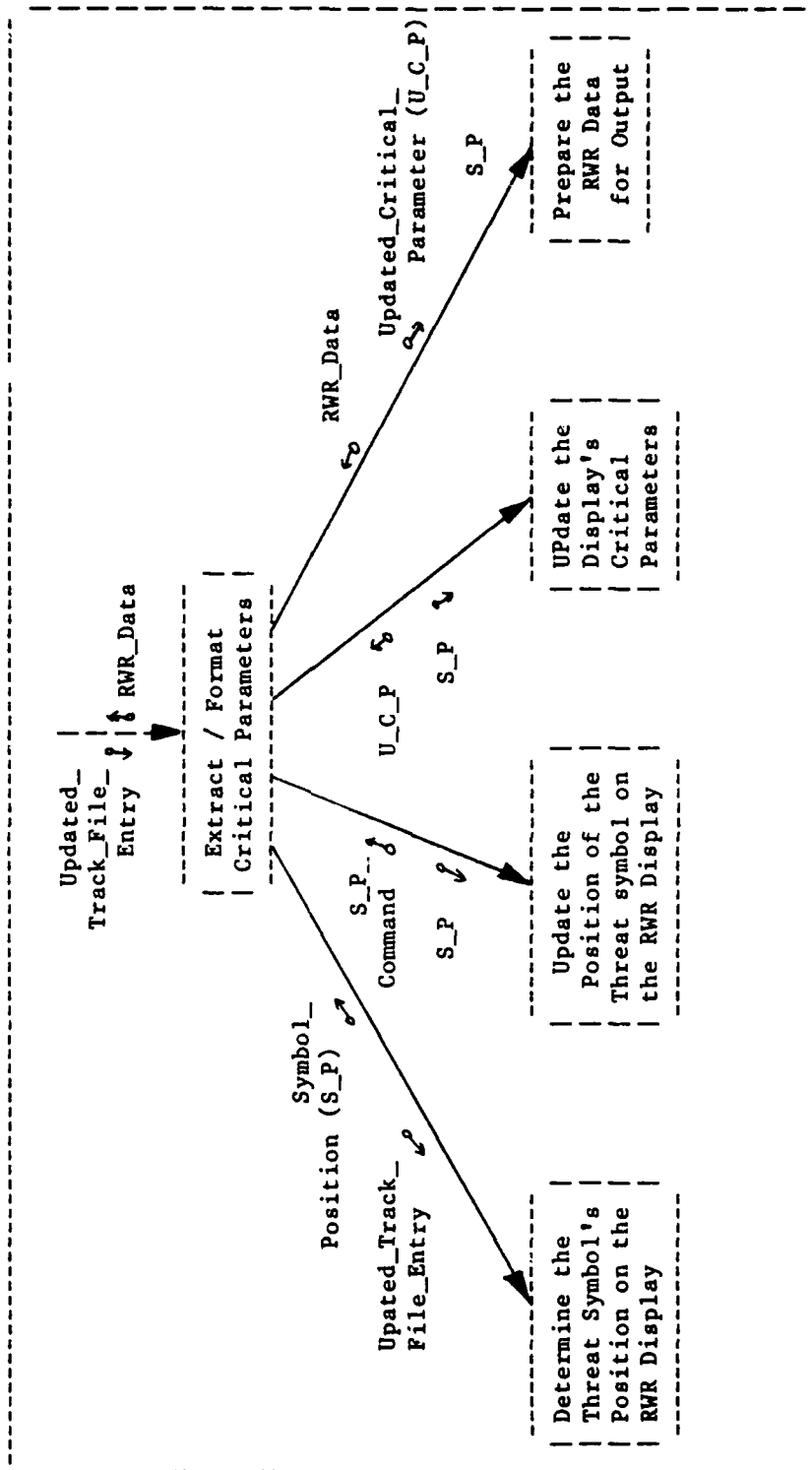


Figure 44. Extract/Format Critical Parameters Structure Chart

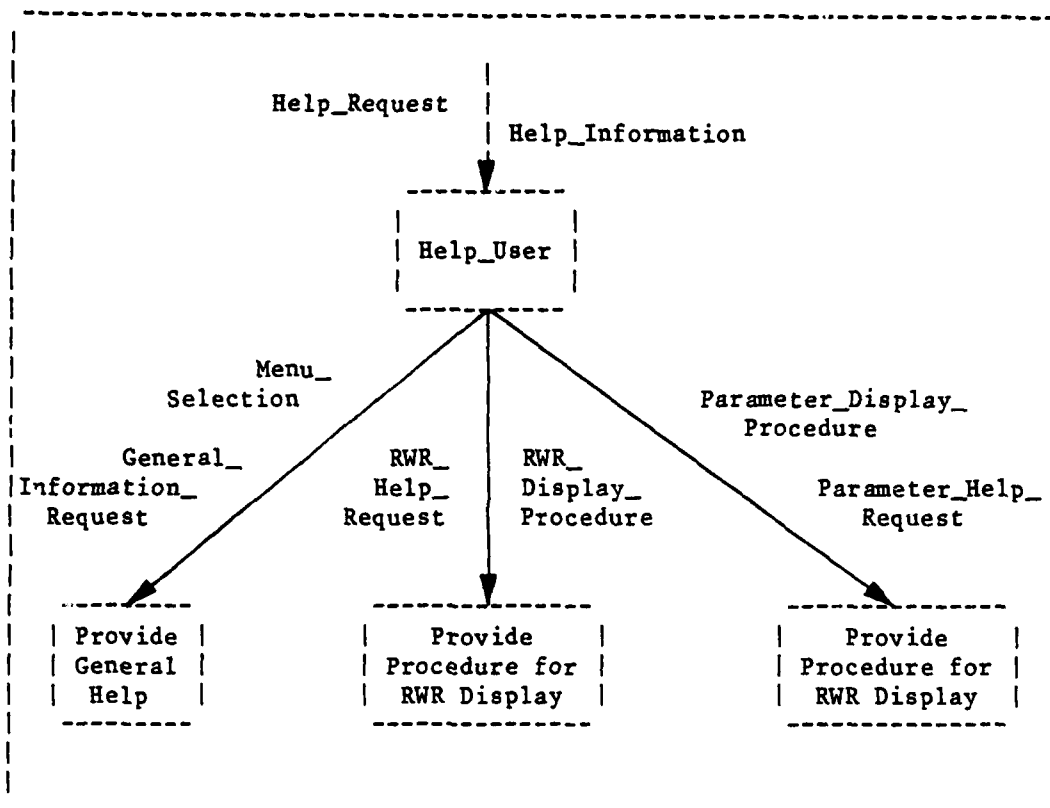


Figure 31. Help User Module Structure Chart

A P P E N D I X E

CGS Warnier-Orr Diagrams and Source Code

This Appendix contains the CGS Warnier-Orr diagrams and the source code compiled listings. The W/O diagrams were used to develop the source code. All of the procedure names referenced in the W/O diagrams are the same as the procedure names used in the source code. A list is the CGS procedure names as well as the page number of each procedure's W/O diagram and source is shown on page 122.

	W/O Chart Page	Source code Page
ALR46CGS	123	148
OPERATOR_INTERFACE	124	151
HELP	125	152
PARAMETER	126	178
CREATE_TRACK_FILE_ENTRIES	127	152
INITIALIZE_TRACK_FILE_ENTRY	128	152
READ_SYMBOL_TYPE_FILE	129	151
SET_STATE_OF_VT100	130	180
GENERATE_FORM	131	180
POSITION_CURSOR	132	179
UPDATE_TRACK_FILE_ENTRY	133	154
PROCESS_ONE_WORD	134	157
SHIFT	135	156
ALR46_DEGREES	136	156
OUTPUT_CRITICAL_PARAMETERS	137	182
POSTION_CURSOR	132	179
RWR_DISPLAY	138	164
CREATE_TRACK_FILE_ENTRIES	127	152
INITIALIZE_TRACK_FILE_ENTRY	128	152
READ_SYMBOL_TYPE_FILE	129	151
INITIALIZE_POSITION	139	176
SET_STATE_OF_4027	140	166
UPDATE_TRACK_FILE_ENTRY	133	154
PROCESS_ONE_WORD	134	157
SHIFT	135	156
ALR46_DEGREES	136	156
OUTPUT_TO_RWR_DISPLAY	141	167
DRIVE_PILOTS_DISPLAY	142	168
OUTPUT_A_SYMBOL	143	169
SEND_CRITICAL_PARAMETERS	144	172
GET_NEXT_COLOR	145	167
X_COORDINATE	146	173
Y_COORDINATE	147	173

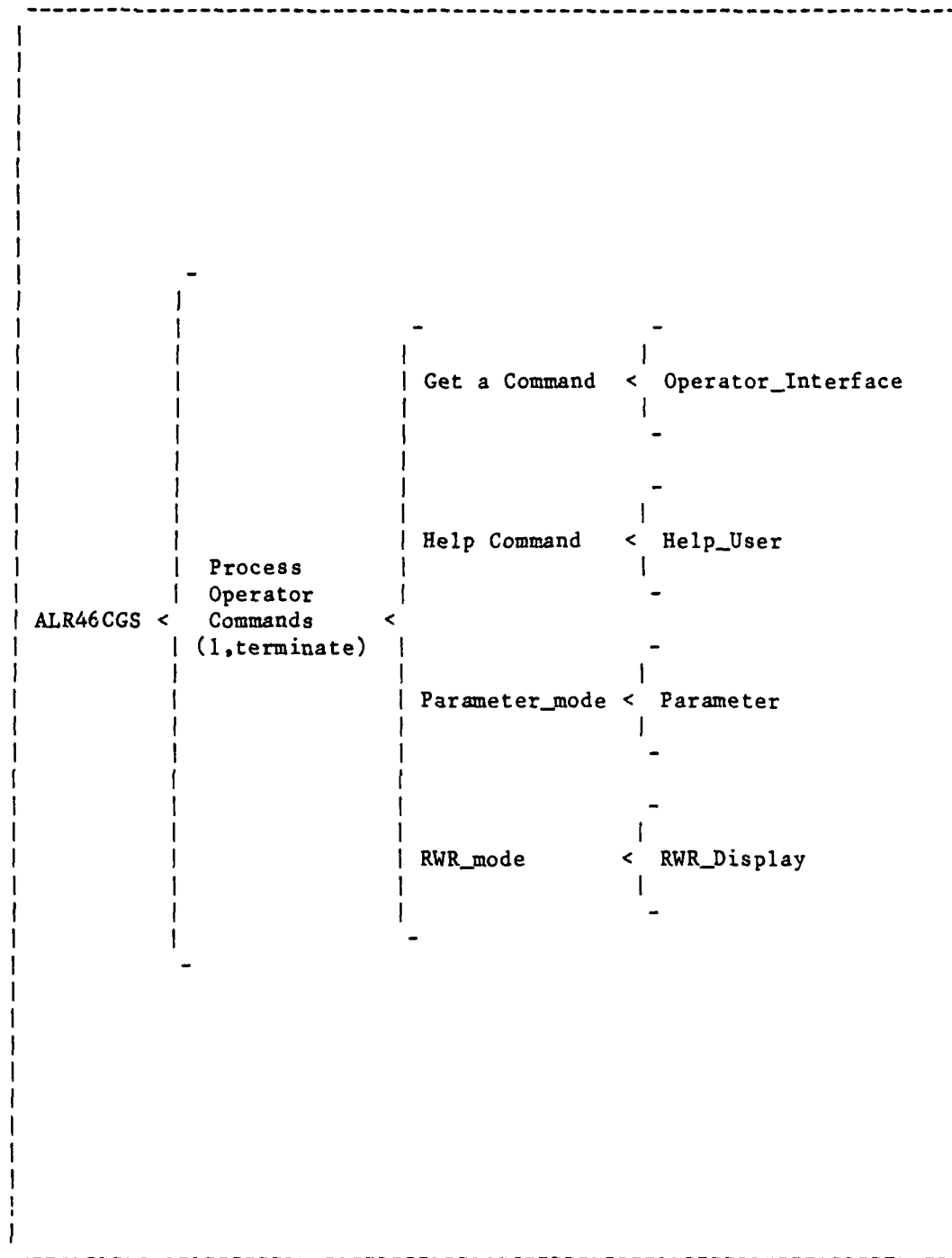
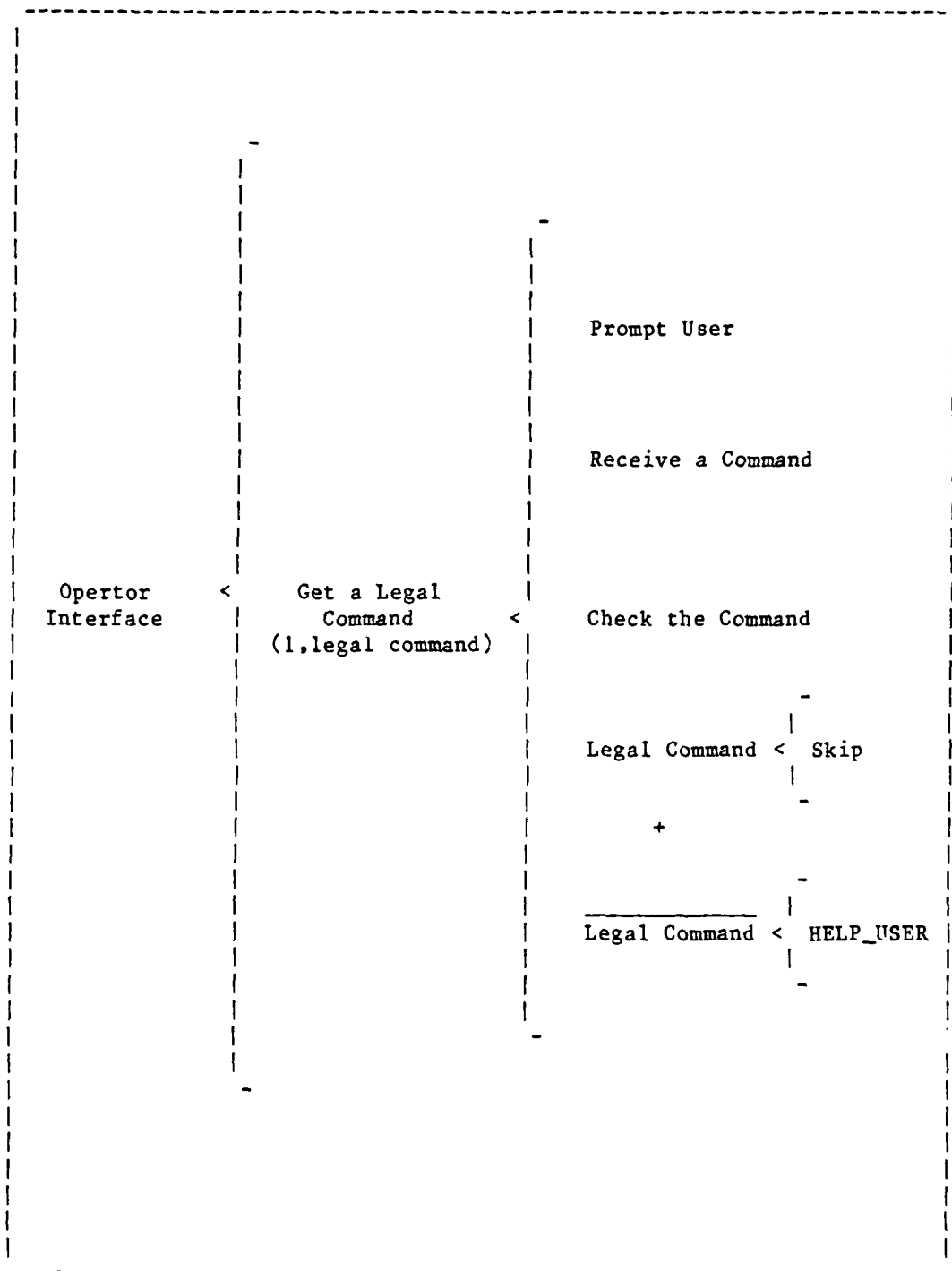
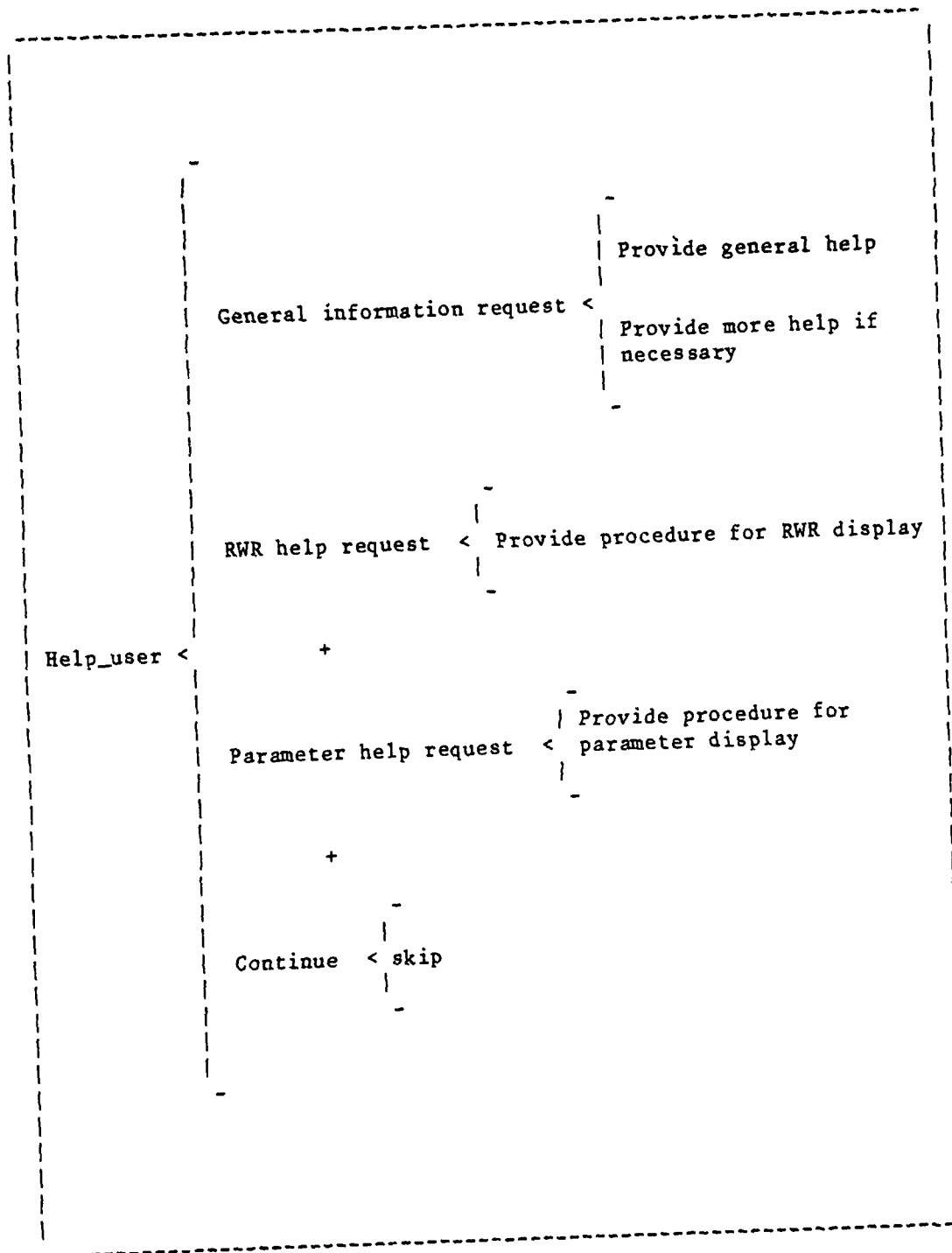


Figure 35. Warnier-Orr System Overview





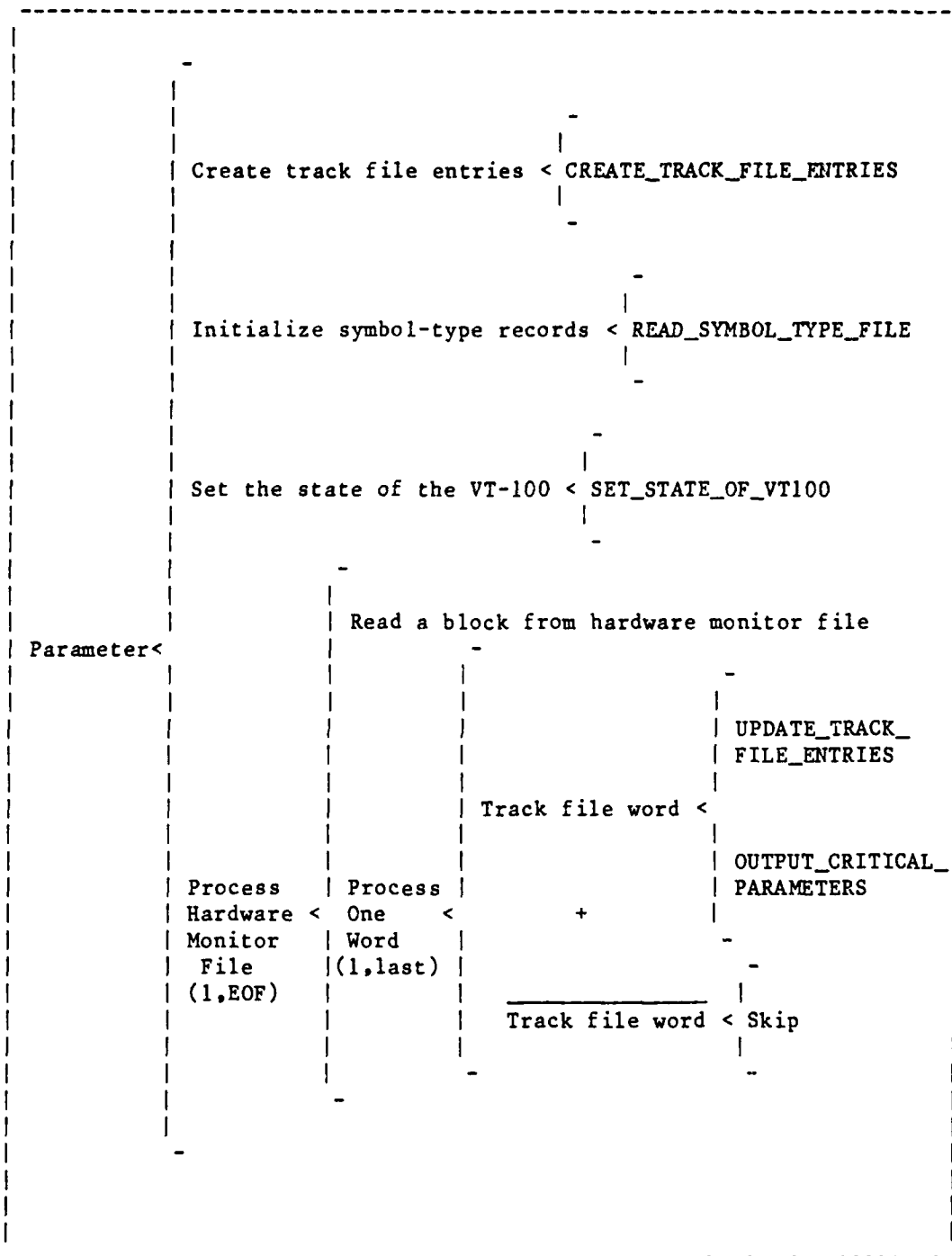


Figure 36. Warnier-Orr Diagram of Parameter Process

CREATE_TRACK_FILE_ENTRY

<

INITIALIZE_TRACK_FILE_ENTRY

Create a new track file entry

Link the new entry into the CGS
linked list

INITIALIZE_TRACK_FILE_ENTRY

<

Set each entry of the
CGS track file to a
known value

READ_SYMBOL_TYPE_FILE

<

Read each pair
of symbol-type
(1.EOF)

<

Read one record

Store the record

SET_STATE_OF_VT100 <

Set CRT into 24 line mode

Set CRT into 132 column mode

Define scroll region between lines 22 and 24

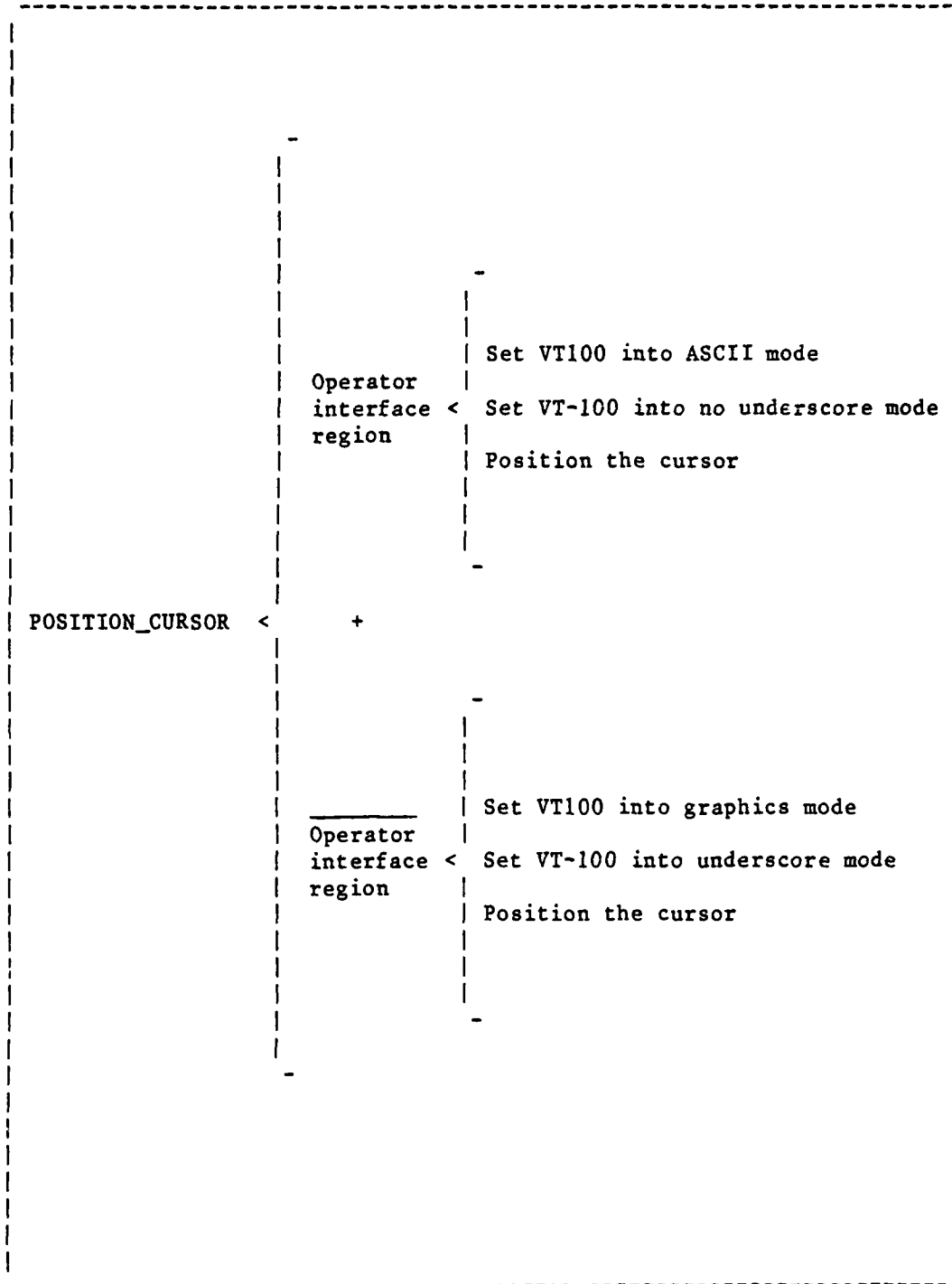
Generate a blank critical parameter form < GENERATE_FORM

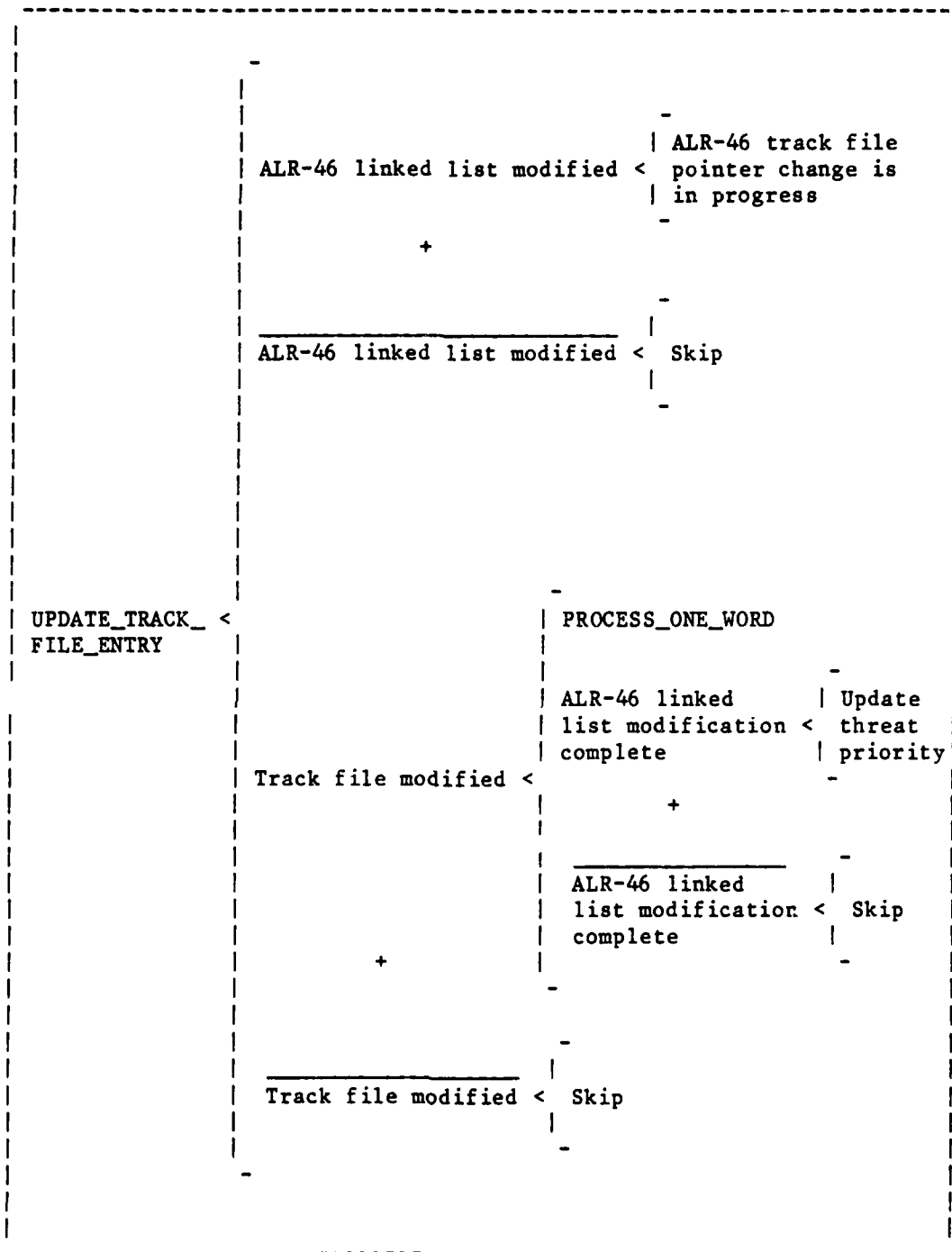
GENERATE_FORM

<

POSITION_CURSOR

Output the form to the VT-100





		-	
			Decode word number zero
			+
			Decode word number two
			+
			Decode word number three
			+
			Decode word number four
			+
			Decode word number five
			+
			Decode word number six
			+
			Decode word number seven
			+
PROCESS_ONE_WORD	<		Decode word number eight
			+
			Decode word number ten
			+
			Decode word number twelve
			+
			Decode word number thirteen
			+
			Decode word number fourteen
			+
			Decode word number fifteen
			-

SHIFT < Extract the bits requested and
return them right justified

```

-
|
| X and Y > 0 and Degrees < 45 < Update angle
|
-
+
-
|
| X and Y > 0 and Degrees > 45 < Update angle
|
-
+
-
|
| ALR-46_DEGREES < X and Y < 0 < Update angle
|
-
+
-
|
| X < 0 and Y >= 0 < Update angle
|
-
+
-
|
| X >= 0 and Y < 0 < Update angle
|
-

```

OUTPUT_
CRITICAL_ <
PARAMETERS

Change to ALR-46 track file < Output new priority

+

Change to ALR-46 track file < Skip

Non_Active_Emitters < Delete all entries from
the display for any threat
no longer active

+

Non_Active_Emitters < Skip

Modified Emitters < Upate the diplay of any
active threat whose parameters
have changed

+

Modified Emitters < Skip

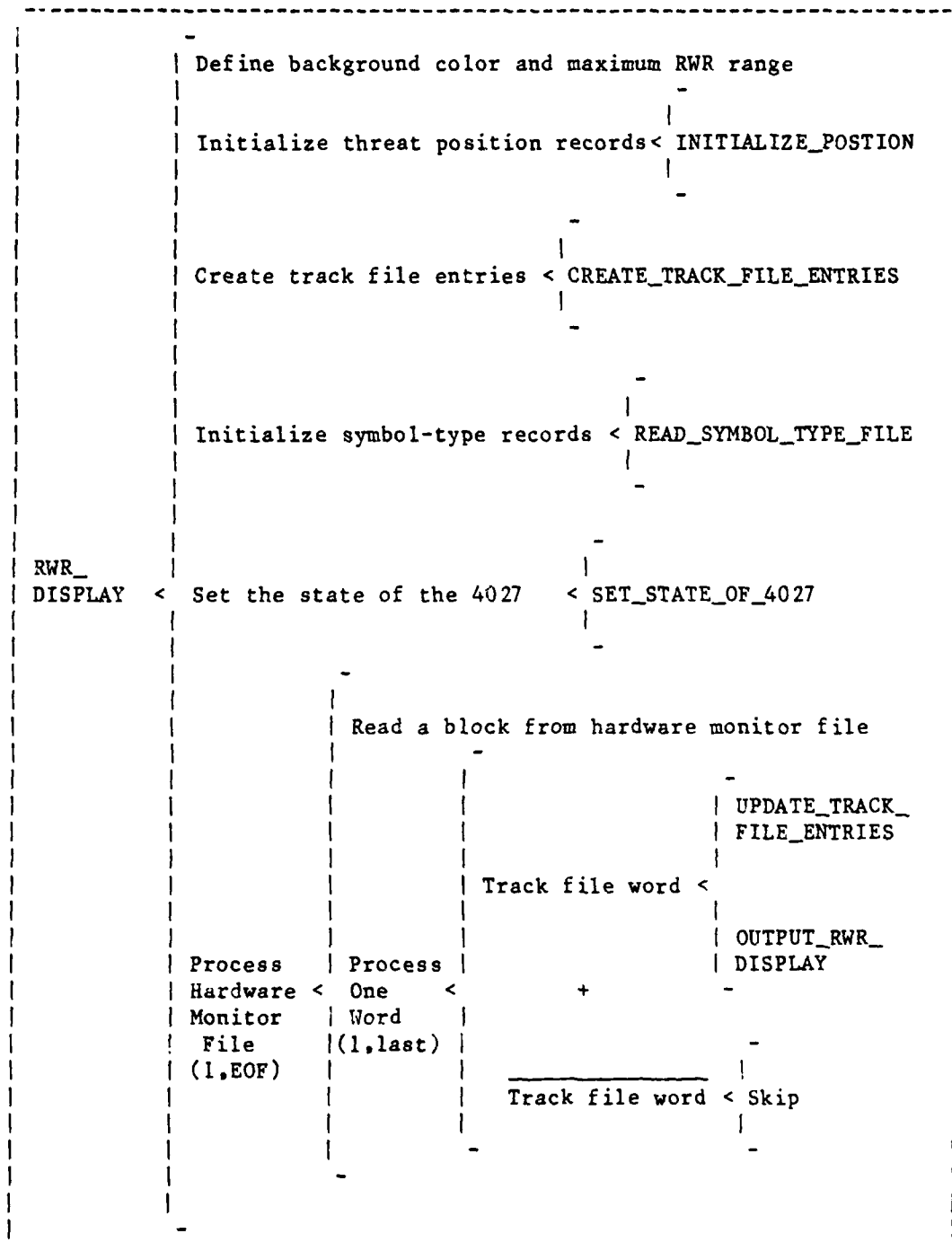


Figure 37. Warnier-Orr Diagram of RWR Process

INITIALIZE_POSITION < Initialize each of the 16 position records

SET_STATE_OF_4027 < Define the operator interface as lines 27-34
Define the graphics interface as lines 1-26
Erase the graphics area
Draw the three RWR range rings
Draw the cross on the pilot's display
Clear the parameter area of the display

OUTPUT_TO_ <
RWR_DISPLAY

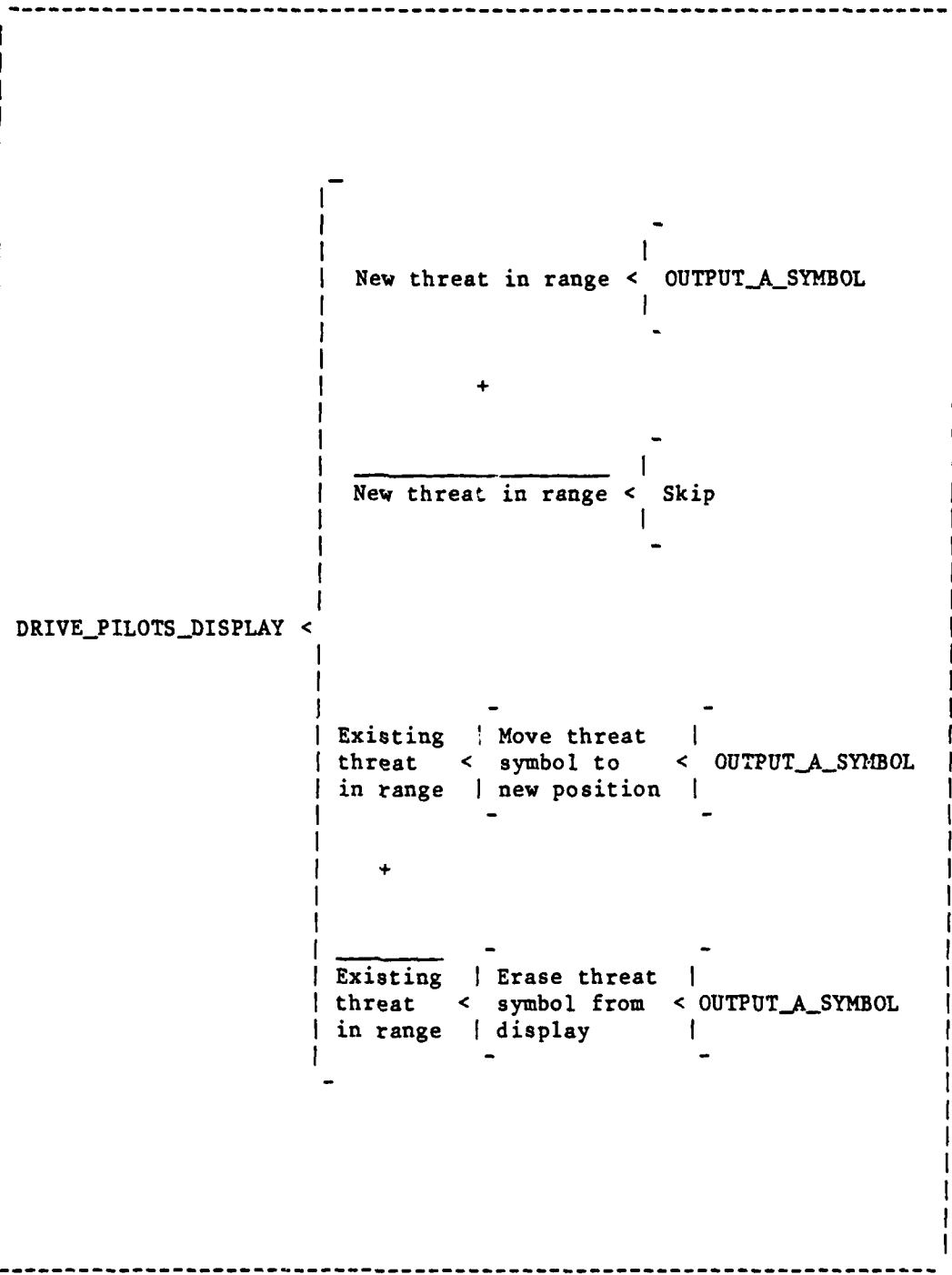
Non Active Emitters < | Delete all information being
| displayed for any threat no
| longer active

+

Non Active Emitters < | Skip

Modified < | Update the display | GET_NEXT_COLOR
emitters < | of any active threat |
| whose range, azimuth, < | DRIVE_PILOTS_
| or symbol parameters | DISPLAY
| have changed |
+ | SEND_CRITICAL_
| PARAMETERS

Modified < | Skip
emitters |



OUTPUT_A_SYMBOL < - Set the color for the next symbol
Position the 4027 graphics cursor
Output the appropriate vector commands
to draw the requested symbol -

SEND_CRITICAL_ <
PARAMETERS

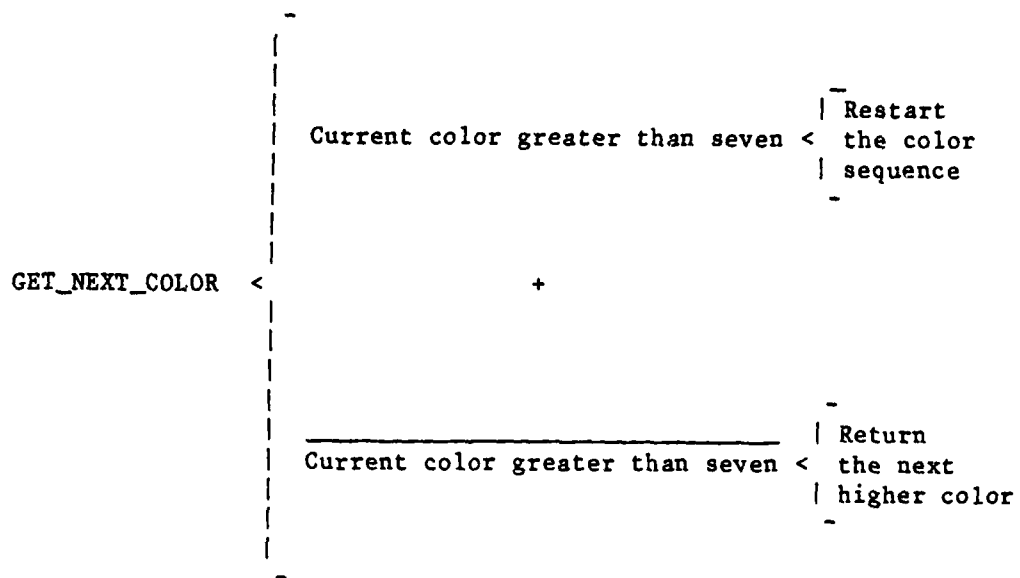
Parameters being deleted <

-
| Send a "blank"
| line to the
| appropriate entry
-

+

Parameters being deleted <

-
| Send the parameters
| to the appropriate
| entry
-



X_COORDINATE <

Generate a scale factor based on the azimuth
and the maximum RWR range

Generate the X coordinate based on the
scale factor and the maximum number of
positions from the origin to the outside
ring on the 4027

Y_COORDINATE <

Generate a scale factor based on the azimuth
and the maximum RWR range

Generate the Y coordinate based on the
scale factor and the maximum number of
positions from the origin to the outside
ring on the 4027

ALR46CGS 10-DEC-1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
 11-NOV-1981 19:45:54 DHA1:[THAMES]ALR46CGS.PAS;309

LINE NUMBERS	LEVEL	PROC STMT	STATEMENT.
1	1	PROGRAM	ALR46CGS (INPUT,OUTPUT,ALR46_DATA,SYMBOL_TYPE_FILE);
2	1		
3			{The ALR-46 Computer Graphics System (CGS) provides the ALR-46 test engineer
4			with a means of analyzing and displaying data taken in real time from the
5			ALR-46 Flight Processor by the hardware monitor, CGS has two modes of
6			operation, the RWR mode and the Parameter mode. The RWR mode simulates the
7			ALR-46 pilot's display, the Tektronix 4027 CRT is used in the RWR mode as the
8			display device. The threat symbols are displayed in color on the display at
9			the correct range and azimuth. The Parameter mode provides the operator with
10			the critical parameters needed to make changes to the Operational Flight
11			Program (OFF).
12			The ALR46CGS program is an executive for CGS. Its function is to call the
13	1		routine to process teh command entered by the operator.}
14	1		
15	1	Const	
16	1		MAXIMUM_NUMBER_OF_EMITTERS = 16;
17	1		MAXIMUM_SYMBOL_PAIRS = 25;
18	1		
19	1	Type	
20	1		ALR46ARRAY = ARRAY [5000..5256] OF INTEGER;
21	1		ADDSTATUS = (IN_PROGRESS,COMPLETED,NONE,IN_PROGRESS);
22	1		ARRAY17 = ARRAY [1..17] OF INTEGER;
23	1		ARRAY4000 = ARRAY [1..4000] OF INTEGER;
24	1		TYPE_HIGHLIGHTER = (NO_HIGHLIGHT,STEADY,CIRCLE,BLINK,CIRCLE_DIAMOND,DOT);
25	1		TYPE_SYMBOL = (SA_1,SA_2,SA_3,SA_4,SA_5,SA_6,SA_7,SA_8,SA_9,HAWK,BAR_2,CROT,WING,
26	1		ALR,AAA_0,AAA_1,AAA_2,AAA_3,STAF2,STAF3,STAF8,UNK0,UNK1,
27	1		UNK2,UNK3);

10-DEC 1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
 11-NOV 1981 19:45:54 IMAGE:THANKS3146/G3.PAS3309

ALR46COS

```

28 1 TYPE_EMITTER = SET OF 1..16;
29 1 TYPE_MODIFIED = SET OF 1..21;
30 1 EMITTER_PTR = ^TRACK_FILE_ENTRY;
31 1 TYPE_PULSE = (JIT,STAG,STAR);
32 1 TYPE_SPECIAL = (WHYY);
33 1 COMMAND_LIST = (HELP,CRITICAL,SWR,OFF,TERMINATE);
34 1 COMMAND_SET = SET OF COMMAND_LIST;
35 1 TRACK_FILE_ENTRY = RECORD
36 1   NEXT_EMITTER : EMITTER_PTR;
37 1   UPDATED_ENTRIES : TYPE_MODIFIED;
38 1   SYMBOL : TYPE_SYMBOL;
39 1   EMITTER_TYPE : INTEGER;
40 1   RF_BAND : INTEGER;
41 1   PRI_FRAME_PERIOD : REAL;
42 1   STAGGER_LEVEL : INTEGER;
43 1   PULSE_TRAIN_DESCR : TYPE_PULSE;
44 1   PRI_AVERAGE : REAL;
45 1   DEVIATION : REAL;
46 1   SPECIAL_FIELD : TYPE_SPECIAL;
47 1   POWER_PGROUND : INTEGER;
48 1   HI_BAND_SIN : INTEGER;
49 1   HI_BAND_COS : INTEGER;
50 1   CEPC : INTEGER;
51 1   HI_BAND_AGE_CNT : INTEGER;
52 1   CD_AGE_COUNTER : INTEGER;
53 1   PRIORITY : INTEGER;
54 1   PRIO_PGROUND : INTEGER;
55 1   AZIMUTH : REAL;
56 1   LETHALITY_RING : REAL;
57 1   ML_AGE : INTEGER;
58 1   RECORD_NUMBER : INTEGER;
59 1   RANGE : REAL;

```

10-DEC-1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
 11-NOV-1981 19:45:54 DMAIL:ETHANESJALR46CGS.PAS:309

ALR46CGS

```

60 1 X_DISPLACEMENT ; REAL;
61 1 Y_DISPLACEMENT ; REAL;
62 1 HIGH_LIGHTER ; TYPE_HIGHLIGHTER;
63 1 END;
64 1
65 1 SYMBOL_TYPE_DATA = RECORD
66 1   SYMBOL_1 : TYPE_SYMBOL;
67 1   SYMBOL_2 : TYPE_SYMBOL;
68 1   THREAT_TYPE ; INTEGER;
69 1 END;
70 1 SYMBOL_TYPE_STRUCTURE = ARRAY [1..MAXIMUM_SYMBOL_PAIRS] OF SYMBOL_TYPE_DATA;
71 1
72 1
73 1
74 1 VAR
75 1   ADDRESS_STATUS : ADDRESS;
76 1   FIRST_AVAILABLE_FIRST_EMITTER, CURRENT_EMITTER : EMITTER_PTR;
77 1   LIST_OF_LEGAL_COMMANDS : COMMAND_SET;
78 1   COMMAND : COMMAND_LIST;
79 1   STOP_COMMAND : BOOLEAN;
80 1   START_ADDRESS_OF_TRACK_FILE : INTEGER;
81 1   END_ADDRESS_OF_TRACK_FILE : INTEGER;
82 1   IMAGE_OF_ALR46_TRACK_FILE : ALR46ARRAY;
83 1   ALR46_FREE_POINTER : INTEGER;
84 1   ALR46_USED_POINTER : INTEGER;
85 1   TRACK_FILE_FREE_POINTER : INTEGER;
86 1   TRACK_FILE_USED_POINTER : INTEGER;
87 1   MONITOR_DATA : ARRAY4000;
88 1   ALR46_DATA : TEXT;
89 1   SYMBOL_TYPE_FILE : TEXT;
90 1   MODIFIED_EMITTERS : TYPE_EMITTER;
91 1   ACTIVE_EMITTERS : TYPE_EMITTER;
92 1   NON_ACTIVE_EMITTERS : TYPE_EMITTER;

```

ALK46CGS 10-DEC-1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
11-NOV-1981 19:45:54 DMAT:CTHAMESJALR46CGS.PAS:309

```

92 1 OLD_EMITTERS : TYPE_EMITTER;
93 1 EMITTER_COUNT : INTEGER;
94 1 EMITTER_PRIORITY : ARRAY[7;
95 1 VALUE
96 1 LIST_OF_LEGAL_COMMANDS := [HELP .. TERMINATE];
97 1 STOP_COMMAND := FALSE;
98 1 START_ADDRESS_OF_TRACK_FILE := 5000;
99 1 END_ADDRESS_OF_TRACK_FILE := 5256;
100 1 TRACK_FILE_FREE_POINTER := 44;
101 1 TRACK_FILE_USED_POINTER := 45;
102 1 MODIFIED_EMITTERS := [];
103 1 ACTIVE_EMITTERS := [];
104 1 NON_ACTIVE_EMITTERS := [];
105 1 OLD_EMITTERS := [];
106 1 MONITOR_DATA := (4000 OF 5000);
107 1 EMITTER_COUNT := 0;
108 1 ADDRESS_STATUS := NONE_IN_PROGRESS;
109 1
110 1
111 1
112 2 {PROCEDURE READ_SYMBOL_TYPE_FILE (VAR SYMBOL_TYPE : SYMBOL_TYPE_STRUCTURE);
113 2 {This procedure reads the symbol-type data from the disk into an internal structure;}
114 2
115 2 CONST
116 2 MAXIMUM_SYMBOL_PAIRS = 25;
117 2
118 2 VAR
119 2 CURRENT_SYMBOL_TYPE : SYMBOL_TYPE_DATA;
120 2 SYMBOL_COUNT : INTEGER;
121 2
122 2 BEGIN
123 2 {BEGIN READ_SYMBOL_TYPE_FILE}

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29	
	11-NOV-1981	19:45:54	DMA1:ETHAMESJALR46CGS.FAS:309	


```

124 0 OPEN (SYMBOL_TYPE_FILE,'(THAMES)SYMBOL.TYP,INT',OLD);      {OPEN THE SYMBOL TYPE FILE}
125 1 RESET (SYMBOL_TYPE_FILE);
126 1 WITH CURRENT_SYMBOL_TYPE DO
127 1 BEGIN
128 2 FOR SYMBOL_COUNT := 1 TO MAXIMUM_SYMBOL_PAIRS DO
129 2 BEGIN
130 3 READLN (SYMBOL_TYPE_FILE,SYMBOL_1,SYMBOL_2,THREAT_TYPE);
131 3 SYMBOL_TYPE [SYMBOL_COUNT] := CURRENT_SYMBOL_TYPE;
132 3 END;
133 2 END; {END OF WITH}
134 1 CLOSE (SYMBOL_TYPE_FILE)
135 1 END;
                                     {END OF READ_SYMBOL_TYPE_FILE}

136 0
137 0
138 2 PROCEDURE HELPUSER;
139 2 BEGIN
140 0 END;
141 0
142 0
143 2 PROCEDURE CREATE_TRACK_FILE_ENTRIES (VAR FIRST_EMITTER,CURRENT_EMITTER ;EMITTER_PTR);
144 2
145
146 2 {This procedure creates the entries in the CGS link list used to store the
147 2 critical parameter data. Each entry is initialized as it is created.}
148 2 VAR
149 2 PREVIOUS_EMITTER : EMITTER_PTR;
150 2 ENTRIES : INTEGER;
151 2
152 3 PROCEDURE INITIALIZE_TRACK_ENTRY (VAR CURRENT_EMITTER ; EMITTER_PTR);
153 3
154 {This procedure initializes each entry in the CGS link list as it is

```

ALR46CGS 10-DEC-1931 18:08:39 VAX-11 PASCAL VERSION V1.1-29
 11-NOV-1981 19:45:54 DWA1:CTH4P*5JALR46CGS.PAS:309

```

155 3      created.)
156 3
157 3
158 0      {EMITTER_COUNT := EMITTER_COUNT + 1;
159 1      WITH CURRENT_EMITTER DO
160 1      BEGIN
161 2      NEXT_EMITTER := NIL;
162 2      UPDATED_ENTRIES := [];
163 2      SYMBOL := SA_2;
164 2      EMITTER_TYPE := EMITTER_COUNT;
165 2      RF_BAND := 2;
166 2      PRI_FRAME_PERIOD := 3.3;
167 2      STAGGER_LEVEL := 4;
168 2      PULSE_TRAIN_DESCR := STAG;
169 2      PRI_AVERAGE := 5.0;
170 2      DEVIATION := 6.6;
171 2      SPECIAL_FIELD := YY;
172 2      POWER_PGROUND := 7;
173 2      HI_BAND_SIN := 8;
174 2      HI_BAND_COS := 9;
175 2      CEPC := 1;
176 2      HI_BAND_AGE_CNT := 2;
177 2      CD_AGE_COUNTER := 3;
178 2      PRIORITY := 4;
179 2      PRI0_PGROUND := 5;
180 2      AZIMUTH := 6.0;
181 2      LETHALITY_RING := 7.7;
182 2      ML_AGE := 8;
183 2      RECORD_NUMBER := 9;
184 2      RANGE := 11.1;
185 2      X_DISPLACEMENT := 1.0;
186 2      Y_DISPLACEMENT := 2.0;
  
```

{INITIALIZE_TRACK_ENTRY}

ALR46CS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DATA1:ETHAMESIALR46CS.PAS;509


```

187      2      HIGH_LIGHTER := DIAMOND;
188      2      END; {END OF WITH}
189      1      END;
190      0
191      0
192      0
193      0      BEGIN
194      0      NEW (CURRENT_EMITTER);
195      1      PREVIOUS_EMITTER := CURRENT_EMITTER;
196      1      INITIALIZE_TRACK_ENTRY (CURRENT_EMITTER);
197      1      FIRST_EMITTER := CURRENT_EMITTER;
198      1
199      1      FOR ENTRIES := 2 TO 16 DO
200      1      BEGIN
201      2      NEW (CURRENT_EMITTER);
202      2      INITIALIZE_TRACK_ENTRY (CURRENT_EMITTER);
203      2
204      2      PREVIOUS_EMITTER := CURRENT_EMITTER;
205      2      PREVIOUS_EMITTER := CURRENT_EMITTER
206      2      END;
207      1      END;
208      0
209      0
210      0
211      0
212      2      PROCEDURE UPDATE_TRACK_FILE_ENTRY (VAR MONITOR_DATA : ARRAY400;
213      2      VAR CURRENT_EMITTER : EMITTER_PTR;
214      2      VAR FIRST_EMITTER : EMITTER_PTR;
215      2      VAR NON_ACTIVE_EMITTERS : TYPE_EMITTER;
216      2      VAR MODIFIED_EMITTERS : TYPE_EMITTER;
217      2      VAR OLD_EMITTERS : TYPE_EMITTER;
218      2      VAR ACTIVE_EMITTERS : TYPE_EMITTER;

```

{END OF INITIALIZE_TRACK_ENTRY}

 {BEGIN CREATE_TRACK_FILE_ENTRY}
 {CREATE A NEW THREAT FILE ENTRY}
 {THIS IS THE FIRST ENTRY}

 {CREATE A NEW THREAT FILE ENTRY}

 {LINK THE NEW ENTRY INTO THE LINK LIST}

 {END OF FOR}
 {END OF CREATE_TRACK_FILE_ENTRIES}

 {PROCEDURE UPDATE_TRACK_FILE_ENTRY}

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMAL:CTHAMESJALR46CGS.PAS:309


```

VAR ADDRESS_STATUS : ADDRESS;
VAR EMITTER_PRIORITY : ARRAY[17];
VAR IMAGE_OF_ALR46_TRACK_FILE : ALR46ARRAY;
VAR ALR46_FREE_POINTER : INTEGER;
VAR ALR46_USED_POINTER : INTEGER;
VAR START_ADDRESS_OF_TRACK_FILE : INTEGER;
VAR ALR46_DATA_INDEX : INTEGER;
VAR SYMBOL_TYPE : SYMBOL_TYPE_STRUCTURE;

{This procedure is the executive to the process of updating the track file
entry. It is used to update the CGS track file each time the ALR-46 changes
its track file. Each word from the hardware monitor is decoded to determine
the treat and the critical parameters affected. The words also determine the
threat's priority.}

CONST
  MAXIMUM_VALUE_FOR_A_BYTE = 20177;

VAR
  BITS_0_3 : INTEGER;
  BIT_1 : INTEGER;
  BIT_5 : INTEGER;
  BITS_6_12 : INTEGER;
  BITS_8_10 : INTEGER;
  BIT_9 : INTEGER;
  DATA : INTEGER;
  EMITTER_NUMBER : INTEGER;
  WORD_NUMBER : INTEGER;
  LAST_EMITTER : BOOLEAN;
  EMITTER_ADDRESS : INTEGER;
  INDEX : INTEGER;
  N : INTEGER;
219 2
220 2
221 2
222 2
223 2
224 2
225 2
226 2
227 2
228
229
230
231
232 2
233 2
234 2
235 2
236 2
237 2
238 2
239 2
240 2
241 2
242 2
243 2
244 2
245 2
246 2
247 2
248 2
249 2
250 2

```

ALR46COS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMH1:([THAMES]ALR46COS.FAS)309


```

251 2
252 2
253 3
254 3
255
256 3
257 3
258 3
259 3
260 3
261 0
262 1
263 1
264 0
265 0
266 3
267 3
268
269 3
270 3
271 3
272 3
273 3
274 0
275 1
276 1
277 1
278 1
279 1
280 1
281 1
282 1

```

```

FUNCTION SHIFT (DATA::MINIMUM_BIT,MAXIMUM_BIT::INTEGER) : INTEGER;
    {This function is used to extract specific bits from a word. The requested
    bits are returned right justified.}

    VAR
        INTERMEDIATE : INTEGER;
    BEGIN
        INTERMEDIATE := DATA MOD 2**(MAXIMUM_BIT + 1); {ZERO THE UNWANTED MSB'S}
        SHIFT := INTERMEDIATE DIV 2**MINIMUM_BIT {RIGHT JUSTIFY THE RESULT}
    END;

FUNCTION ALR46_DEGREES(VAR X,Y : REAL) : REAL;
    {This function converts the X and Y displacement value received from the
    ALR-46 into azimuth. Refer to Appendix B for additional information.}

    VAR
        DEGREES : REAL;
    BEGIN
        IF Y = 0.0 THEN {WE CAN NOT DIVIDE BY ZERO}
            Y := 1.0E-20;
        DEGREES := ABS (ARCTAN(X / Y)) * (360.0 / (2.0 * 3.14159));
        {CONVERT THE ANGLE TO THE ALR-46 COORDINATES}

        IF (X >= 0.0) AND (Y >= 0.0) AND (DEGREES < 45.0) THEN
            DEGREES := 360.0 - (45.0 - DEGREES)
        ELSE

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:[THAMESIALR46CGS.FAS;309

```

283 1 IF (X >= 0.0) AND (Y >= 0.0) AND (DEGREES >= 45.0) THEN
284 1 DEGREES := DEGREES - 45.0
285 1 ELSE
286 1 IF (X >= 0.0) AND (Y < 0.0) THEN
287 1 DEGREES := 45.0 + (90.0 - (DEGREES))
288 1 ELSE
289 1 IF (X < 0.0) AND (Y < 0.0) THEN
290 1 DEGREES := 135.0 + DEGREES
291 1 ELSE
292 1 IF (X < 0.0) AND (Y >= 0.0) THEN
293 1 DEGREES := 225.0 + (90.0 - (DEGREES))
294 1 ELSE
295 1 WRITELN ('***ALR46_DEGREES*** BAD X Y DATA');
296 1 ALR46_DEGREES := DEGREES;
297 1 END;
298 0
299 0
300 0
301 0
302 0
303 3 PROCEDURE PROCESS_ONE_WORD(VAR CURRENT_EMITTER : EMITTER_PTR; {PROCEDURE PROCESS_ONE_WORD}
304 3 VAR DATA : INTEGER;
305 3 VAR ADDRESS_STATUS : ADDRESS;
306 3 VAR SYMBOL_TYPE : SYMBOL_TYPE_STRUCTURE);
307 3
308 3 {This procedure decodes a word from the hardware monitor and updates the
309 3 appropriate critical parameters.}
310 3
311 3 CONST
312 3 STARTING_ADDRESS_OF_ALR46_SYMBOL_TABLE = 8000;
313 3
314 3 VAR

```

(RETURN THE ALR-46 ANGLE)
(END OF ALR-46 DEGREES)

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:CTHAMESIALR46CGS.PAS:309


```

315 3      CURRENT_SYMBOL_TYPE := SYMBOL_TYPE_DATA;
316 3
317 3      BEGIN
318 0      WITH CURRENT_EMITTER DO
319 1      CASE WORD_NUMBER OF
320 2
321 2      0 : BEGIN
322 3      IMAGE_OF_ALR46_TRACK_FILE [MONITOR_DATAALR46_DATA_INDEX + 1] := DATA;
323 3      ADDRESS_STATUS := IN_PROGRESS
324 3      END;
325 2
326 2      2 : BEGIN
327 3      CURRENT_SYMBOL_TYPE := SYMBOL_TYPE [DATA - STARTING_ADDRESS_OF_ALR46_SYMBOL_TABLE];
328 3      SYMBOL := CURRENT_SYMBOL_TYPE.SYMBOL_1;
329 3      EMITTER_TYPE := CURRENT_SYMBOL_TYPE.THREAT_TYPE;
330 3      UPDATED_ENTRIES := UPDATED_ENTRIES + [1,2]
331 3      END;
332 2
333 2      3 : BEGIN
334 3      PRI_FRAME_PERIOD := DATA / 10.0;
335 3      UPDATED_ENTRIES := UPDATED_ENTRIES + [4]
336 3      END;
337 2
338 2      4 : BEGIN
339 3      HI_BAND_SIN := SHIFT(DATA,8,15);
340 3      UPDATED_ENTRIES := UPDATED_ENTRIES + [10]
341 3      END;
342 2
343 2      5 : BEGIN
344 3      HI_BAND_COS := SHIFT(DATA,8,15);
345 3      UPDATED_ENTRIES := UPDATED_ENTRIES + [11]
346 3      END;

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:(THAMESJALR46CGS.FAS;309

```

347 2
348 2
349 3
350 3
351 4
352 4
353 4
354 4
355 4
356 4
357 4
358 3
359 3
360 3
361 3
362 2
363 2
364 3
365 3
366 4
367 4
368 4
369 4
370 4
371 4
372 3
373 3
374 3
375 3
376 3
377 3
378 3

        6 : BEGIN
          BITS_0_3 := SHIFT(DATA,0,3);
          CASE BITS_0_3 OF
            0 : HIGH_LIGHTER := NO_HIGHLIGHTER;
            1,3 : HIGH_LIGHTER := BLINK_CIRCLE;
            2 : HIGH_LIGHTER := STEADY_CIRCLE;
            4 : HIGH_LIGHTER := DIAMOND;
            8 : HIGH_LIGHTER := DOT;
          OTHERWISE WRITELN('**PROCESS_ONE_WORD** BAD CASE INDEX IN WORD 6');
          END; {CASE}
          PRIORITY := SHIFT(DATA,5,9);
          PRIO_POWEROUND := SHIFT(DATA,10,15);
          UPDATED_ENTRIES := UPDATED_ENTRIES + [15,16]
          END; {END OF 6}

        7 : BEGIN
          BITS_0_3 := SHIFT(DATA,0,3);
          CASE BITS_0_3 OF
            0 : RF_BAND := 0;
            1 : RF_BAND := 1;
            2 : RF_BAND := 2;
            4 : RF_BAND := 3;
            8 : RF_BAND := 4;
          END; {END CASE}
          BIT_9 := SHIFT(DATA,9,9);
          IF BIT_9 = 1 THEN
            PULSE_TRAIN_DESCR := JIT
          ELSE
            IF (BIT_9 = 0) AND (PULSE_TRAIN_DESCR = JIT) THEN
              PULSE_TRAIN_DESCR := STAB;
            POWER_FWROUND := SHIFT(DATA,10,15);
          END;

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:ETHANESJALR46CGS.PAS;309


```

379      UPDATED_ENTRIES := UPDATED_ENTRIES + [3,5,9]
380      END; {END OF 7}
381
382      8 :BEGIN
383          BITS_8_10 := SHIFT(DATA,8,10);
384          IF (BITS_8_10 = 0) AND (PULSE_TRAIN_DESCR <> J11) THEN
385              BEGIN
386                  PULSE_TRAIN_DESCR := STAB;
387                  STAGGER_LEVEL := 0;
388                  END; {END OF THEN}
389              IF BITS_8_10 <> 0 THEN
390                  BEGIN
391                      PULSE_TRAIN_DESCR := STAG;
392                      STAGGER_LEVEL := BITS_8_10 + 1
393                  END; {END OF THEN}
394                  CEPC := SHIFT(DATA,0,2);
395                  HL_AGE := SHIFT(DATA,5,7);
396                  CD_AGE_COUNTER := SHIFT(DATA,11,15);
397                  UPDATED_ENTRIES := UPDATED_ENTRIES + [5,12,19,14]
398                  END; {END OF 8}
399
400      9 :BEGIN
401          BIT_5 := SHIFT(DATA,5,5);
402          BITS_6_12 := SHIFT(DATA,6,12);
403          IF BIT_5 = 1 THEN
404              X_DISPLACEMENT := (MAXIMUM_VALUE_FOR_A_BYTE - BITS_6_12) * (-1)
405          ELSE
406              X_DISPLACEMENT := BITS_6_12;
407          RANGE := SORT (X_DISPLACEMENT **2 + Y_DISPLACEMENT **2);
408          LETHALITY_RING := RANGE / 10000.0;
409          AZIMUTH := ALR46_DEGREES (X_DISPLACEMENT, Y_DISPLACEMENT);
410          UPDATED_ENTRIES := UPDATED_ENTRIES + [17,18,21]

```

{SAVE BITS 8 THROUGH 10}
 {SAVE THE THREE LSB'S}
 {SAVE BITS 5-7}
 {SAVE BITS 11 THROUGH 15}
 {SAVE BIT 5}
 {SAVE BITS 6 THROUGH 12}

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:CTHAMESJALR46CGS.PAS:309

```

411      3      END; {END 9}
412      2
413      2      10:BEGIN
414      3      BIT_5 := SHIFT(DATA,5,5);
415      3      BITS_6_12 := SHIFT(DATA,6,12);
416      3      IF BIT_5 = 1 THEN
417      3      Y_DISPLACEMENT := (MAXIMUM_VALUE_FOR_A_BYTE - BITS_6_12) * (-1)
418      3      ELSE
419      3      Y_DISPLACEMENT := BITS_6_12;
420      3      RANGE := SORT (X_DISPLACEMENT *2 + Y_DISPLACEMENT *2);
421      3      LETHALITY_RING := RANGE / 1000.0;
422      3      AZIMUTH := ALR46_DEGREES (X_DISPLACEMENT, Y_DISPLACEMENT);
423      3      UPDATED_ENTRIES := UPDATED_ENTRIES + [17,18,21]
424      3      END; {END 10}
425      2
426      2      12:BEGIN
427      3      PRI_AVERAGE := DATA / 10.0;
428      3      UPDATED_ENTRIES := UPDATED_ENTRIES + [6]
429      3      END; {END 12}
430      2
431      2      13:BEGIN
432      3      BIT_1 := SHIFT(DATA,0,0);
433      3      IF BIT_1 = 1 THEN
434      3      SPECIAL_FIELD := YY
435      3      ELSE
436      3      SPECIAL_FIELD := NN;
437      3      DEVIATION := SHIFT(DATA,6,15);
438      3      UPDATED_ENTRIES := UPDATED_ENTRIES + [7,8]
439      3      END; {END 13}
440      2
441      2      14:BEGIN
442      3      HI_BAND_AGE_CMT := SHIFT(DATA,3,7);

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DNA1:[THAMES]ALR46CGS.PAS:309


```

443      UPDATED_ENTRIES := UPDATED_ENTRIES + [13]
444      END; {END 14}
445
446      15:BEGIN
447      RECORD_NUMBER := SHIFT(DATA,0,4);
448      UPDATED_ENTRIES := UPDATED_ENTRIES + [20]
449      END; {END 15}
450      OTHERWISE WRITELN('ERROR--PROCEDURE = PROCESS.ONE_WORD')
451      END; {END OF CASE}
452
453      1
454      0
455      0
456      0
457      0
458      1
459      2
460      2
461      2
462      2
463      1
464      1
465      1
466      2
467      2
468      2
469      2
470      1
471      1
472      1
473      1
474      2

```

```

      {SAVE BITS 0 TO 4}

      {END OF PROCESS.ONE_WORD}

      {UPDATE_TRACK_FILE_ENTRY}
      {IF THE ALR-46 LINKED LIST POINTER HAS BEEN CHANGED-THEN}
      IF MONITOR_DATA [ALR46_DATA_INDEX + 1] = TRACK_FILE_FREE_POINTER THEN
      BEGIN
      ALR46_FREE_POINTER := MONITOR_DATA [ALR46_DATA_INDEX];
      ADDRESS_STATUS := IN_PROGRESS
      END {END OF IF}
      ELSE
      {IF THE ALR-46 LINKED LIST POINTER HAS BEEN CHANGED-THEN}
      IF MONITOR_DATA [ALR46_DATA_INDEX + 1] = TRACK_FILE_USED_POINTER THEN
      BEGIN
      ALR46_USED_POINTER := MONITOR_DATA [ALR46_DATA_INDEX];
      ADDRESS_STATUS := IN_PROGRESS
      END
      ELSE
      {IF THE ALR-46 LINKED LIST DATA WORD HAS BEEN CHANGED-THEN}
      IF (MONITOR_DATA [ALR46_DATA_INDEX + 1] >= START_ADDRESS_OF_TRACK_FILE) AND
      (MONITOR_DATA [ALR46_DATA_INDEX + 1] <= END_ADDRESS_OF_TRACK_FILE) THEN
      BEGIN
      { GET THE ENITTER NUMBER }

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:CTHAMESIALR46CGS.PAS:309


```

475      EMITTER_NUMBER := ((MONITOR_DATA[ALR46_DATA_INDEX + 1] -
476      START_ADDRESS_OF_TRACK_FILE) DIV 16) + 1);
477      { POINT TO CURRENT EMITTER }
478      CURRENT_EMITTER := FIRST_EMITTER;
479      IF EMITTER_NUMBER > 1 THEN
480      BEGIN
481          FOR N := 2 TO EMITTER_NUMBER DO
482              CURRENT_EMITTER := CURRENT_EMITTER + NEXT_EMITTER
483      END;
484      { GET THE TRACK FILE ENTRY WORD NUMBER }
485      WORD_NUMBER := (MONITOR_DATA [ALR46_DATA_INDEX + 1] -
486      START_ADDRESS_OF_TRACK_FILE) MOD 16;
487      DATA := MONITOR_DATA [ALR46_DATA_INDEX]; { SHORTEN THE VARIABLE NAME FOR THE WORD }
488      PROCESS_ONE_WORD (CURRENT_EMITTER, DATA, ADDRESS_STATUS, SYMBOL_TYPE);
489      IF WORD_NUMBER <> 0 THEN
490          { WORD 0 IS AN ADDRESS, NOT DATA }
491          MODIFIED_EMITTERS := [EMITTER_NUMBER] + MODIFIED_EMITTERS;
492          IF (WORD_NUMBER > 0) AND (ADDRESS_STATUS = IN_PROGRESS) THEN
493              ADDRESS_STATUS := COMPLETED;
494          END {END IF}
495      ELSE
496          IF ADDRESS_STATUS = IN_PROGRESS THEN
497              ADDRESS_STATUS := COMPLETED;
498          { IF THE ALR-46 LINK LIST ADDRESS CHANGE IS COMPLETE - THEN }
499          IF ADDRESS_STATUS = COMPLETED THEN
500              BEGIN
501                  IF ALR46_USED_POINTER = TRACK_FILE_USED_POINTER THEN
502                      BEGIN
503                          LAST_EMITTER := TRUE;
504                          ACTIVE_EMITTERS := [];
505                          EMITTER_PRIORITY [1] := 0;
506                          END {END OF IF}
507                      ELSE
508                          { WE HAVE NO ACTIVE EMITTERS }
509                      END
510          END
511      
```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:[THAMES]ALR46CGS.PAS:309

```

507      2      BEGIN
508      3      EMITTER_ADDRESS := ALR46_USED_POINTER;
509      3      LAST_EMITTER := FALSE;
510      3      ACTIVE_EMITTERS := [];
511      3      END; {END OF ELSE}
512      2      INDEX := 1;
513      2      WHILE NOT LAST_EMITTER DO
514      2      BEGIN
515      3      EMITTER_PRIORITY [1] := INDEX;
516      3      INDEX := INDEX + 1;
517      3      EMITTER_PRIORITY [INDEX] := (EMITTER_ADDRESS - START_ADDRESS_OF_TRACK_FILE) DIV 16) + 1;
518      3      ACTIVE_EMITTERS := ACTIVE_EMITTERS + EMITTER_PRIORITY [INDEX];
519      3      {GET THE NEXT EMITTER ADDRESS}
520      3      EMITTER_ADDRESS := IMAGE_OF_ALR46_TRACK_FILE [EMITTER_ADDRESS];
521      3      IF EMITTER_ADDRESS = TRACK_FILE_USED_POINTER THEN
522      3      LAST_EMITTER := TRUE
523      3      END; {END WHILE}
524      2      NON_ACTIVE_EMITTERS := OLD_EMITTERS - ACTIVE_EMITTERS;
525      2      OLD_EMITTERS := ACTIVE_EMITTERS
526      2      END; {END OF IF}
527      1      END;
528      0
529      0
530      0
531      2      PROCEDURE RWR_DISPLAY (VAR MONITOR_DATA : ARRAY[4000];
532      2      VAR CURRENT_EMITTER : EMITTER_PTR;
533      2      VAR FIRST_EMITTER : EMITTER_PTR;
534      2      VAR NON_ACTIVE_EMITTERS : TYPE_EMITTER;
535      2      VAR MODIFIED_EMITTERS : TYPE_EMITTER;
536      2      VAR OLD_EMITTERS : TYPE_EMITTER;
537      2      VAR ACTIVE_EMITTERS : TYPE_EMITTER;
538      2      VAR ADDRESS_STATUS : ADDRESS_STATUS;

```

{RANK THE THREATS BY PRIORITY}

{STORE COUNT OF ACTIVE EMITTERS}

{END OF UPDATE_TRACK_FILE_ENTRY}

{PROCEDURE RWR_DISPLAY}

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:ITHMESJALR46CGS.FAS;309


```

539 2      VAR EMITTER_PRIORITY : ARRAY17;
540 2      VAR IMAGE_OF_ALR46_TRACK_FILE : ALR46ARRAY;
541 2      VAR ALR46_FREE_POINTER : INTEGER;
542 2      VAR ALR46_USED_POINTER : INTEGER;
543 2      VAR START_ADDRESS_OF_TRACK_FILE : INTEGER;
544 2
545 2
546
547
548 {This is the executive for the RWR display process. It first sets the VT-100
549 CRT into the mode required for the RWR process. Next it processes a word from
550 the hardware monitor. If the threat's symbol or position is affected, the
551 appropriate changes are made to the display.}
552
553 CONST
554     PC_FLAG = %0177777;
555
556 TYPE
557     TYPE_STATUS = (NEW,UPDATED,REMOVE,OUT_OF_RANGE);
558     RWR_DISPLAY_DATA = RECORD
559         STATUS : TYPE_STATUS;
560         SYMBOL : TYPE_SYMBOL;
561         RANGE : REAL;
562         AZIMUTH : REAL;
563         NEW_X_POSITION : INTEGER;
564         NEW_Y_POSITION : INTEGER;
565         OLD_X_POSITION : INTEGER;
566         OLD_Y_POSITION : INTEGER;
567         COLOR : INTEGER;
568     END;
569     TYPE_LOCATION = ARRAY(1..16) OF RWR_DISPLAY_DATA;
570
571 VAR
572     ALR46_DATA_INDEX : INTEGER;
573     EMITTER_POSITIONS : TYPE_LOCATION;

```

10-DEC-1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
11-NOV-1981 19:45:54 LMA1:ITHAMESIALK46CGS.FAS;309

ALK46CGS

571 2 CURRENT_COLOR : INTEGER;
572 2 BACKGROUND_COLOR : INTEGER;
573 2 MAXIMUM_RWR_RANGE : REAL;
574 2 SYMBOL_TYPE : SYMBOL_TYPE_STRUCTURE;
575 2
576 2
577 2

PROCEDURE SET_STATE_OF_4027(VAR BACKGROUND_COLOR : INTEGER); {PROCEDURE SET_STATE_OF_4027}

{This procedure sets the 4027 CRT into the state required by the RWR display.
For example, the 4027 has to have a graphics region defined for the pilot's
display and a region defined for the operator dialogue.}

VAR
LINE : INTEGER;

BEGIN
589 0 WRITELN ('RWR 27', {DEFINE THE OPERATOR INTERFACE AS LINES 27 - 34}
590 1 'RWR 1,26', {DEFINE THE GRAPHICS REGION AS LINES 1 - 26}
591 1 'RWR 6 C',BACKGROUND_COLOR;1); {ERASE THE GRAPHICS AREA USING BACKGROUND COLOR}
592 1
593 1 WRITELN ('COL C1', {SET THE FOREGROUND COLOR TO C1}
594 1 'RWR 425, 177', {CENTER THE CURSOR IN THE GRAPHICS AREA}
595 1 'RWR 170', {DRAW AN RWR RANGE RING WITH RADIUS OF 170}
596 1 'RWR 85', {DRAW AN RWR RANGE RING WITH RADIUS OF 85}
597 1 'RWR 40', {DRAW AN RWR RANGE RING WITH RADIUS OF 40}
598 1 'RWR 1'); {DRAW AN RWR RANGE RING WITH RADIUS OF 1}
599 1
600 1 {DRAW LINES ON THE RWR DISPLAY}
601 1 WRITELN('RWR 464 177 594 177 RWR 425 216 425 346 RWR 385 177 255 177');
602 1 WRITELN('RWR 424 137 424 7 RWR 517 269 545 297 RWR 332 269 304 297');

```

10-DEC-1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
11-NOV-1981 19:45:54 DMA1:ETHAMESJALR46UGS.PAS;309

ALR46UGS

603 1 WRITELN('I'VEC 332 84 304 56 I'VEC 517 84 545 56');
604 1
605 1 WRITELN('I'VECO,350 I'STR/ENTRY # SYMBOL RANGE AZIMUTH /');{WRITE THREAT HEADER INFORMATION}
606 1 FOR LINE := 0 TO 24 DO {CLEAR THE THREAT STATUS AREA ON THE 4027}
607 1 WRITELN('I'VECO',(LINE # 24):1,'I'STR/
608 1 END; {SET_STATE_OF_4027}
609 0
610 0
611 0
612 0

613 3 PROCEDURE OUTPUT_TO_RWR_DISPLAY (VAR FIRST_EMITTER : EMITTER_PTR;
614 3 VAR MODIFIED_EMITTERS : TYPE_EMITTER;
615 3 VAR NON_ACTIVE_EMITTERS : TYPE_EMITTER;
616 3 VAR EMITTER_POSITIONS : TYPE_LOCATION;
617 3 VAR BACKGROUND_COLOR : INTEGER;
618 3 VAR MAXIMUM_RWR_RANGE : REAL;
619 3 VAR CURRENT_COLOR : INTEGER);
620 3
621 {This procedure is responsible for maintaining the RWR display. It must
622 simulate the Pilot's display and keep the critical parameters current.}
623 3
624 3
625 3
626 3 VAR
627 3 CURRENT_THREAT : RWR_DISPLAY_DATA;
628 3 EMITTER_NUMBER : INTEGER;
629 3 CURRENT_EMITTER : EMITTER_PTR;
630 3
631 3 PROCEDURE GET_NEXT_COLOR(VAR SYMBOL : TYPE_SYMBOL;
632 4 VAR COLOR : INTEGER;
633 4 VAR CURRENT_COLOR : INTEGER;
634 4

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:CTHAMESJALR46CGS.FAS3309


```

635 4          VAR BACKGROUND_COLOR : INTEGER);
636 4
637 4      {This procedure sets the color of the next new threat symbol The 4027 CRT can
638 4      have eight active colors.}
639 4
640 4      BEGIN {GET_NEXT_COLOR}
641 0          IF CURRENT_COLOR > 7 THEN
642 1              BEGIN
643 2                  IF BACKGROUND_COLOR = 0 THEN
644 2                      CURRENT_COLOR := 1
645 2                  ELSE
646 2                      CURRENT_COLOR := 0;
647 2                  {WRITELN ('***GET_NEXT_COLOR*** RESET CURRENT_COLOR TO ',CURRENT_COLOR);}
648 2                  END; {END OF IF}
649 1                  COLOR := CURRENT_COLOR;
650 1                  CURRENT_COLOR := SUCC (CURRENT_COLOR);
651 1                  IF CURRENT_COLOR = BACKGROUND_COLOR THEN
652 1                      CURRENT_COLOR := SUCC (CURRENT_COLOR);
653 1                  {WRITELN ('***GET_NEXT_COLOR*** EXITING COLOR =',COLOR);}
654 1                  END;
655 0
656 0
657 0
658 0      {PROCEDURE DRIVE_PILOTS_DISPLAY (VAR CURRENT_THREAT : RWR_DISPLAY_DATA;
659 4      VAR MAXIMUM_RWR_RANGE : REAL;
660 4      VAR BACKGROUND_COLOR : INTEGER);}
661 4
662 4
663 4      {This procedure maintains the simulation of the display. New symbols must be
664 4      displayed, old symbols must be re-positioned, and deleted symbols must be
665 4      erased from the screen.}
666 4

```

```

10-DEC-1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
11-NOV-1981 19:45:54 DMA1:[THAMES]AIR46CGS.PAS;309

ALR46CGS

667 4
668 5
669 5
670 5
671 5
672 5
673 5
674 5
675 5
676 5
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698

PROCEDURE OUTPUT_A_SYMBOL (VAR SYMBOL : TYPE_SYMBOL; {PROCEDURE OUTPUT_A_SYMBOL}
VAR COLOR : INTEGER;
VAR X : INTEGER;
VAR Y : INTEGER);

{This procedure draws a threat's symbol on the Pilot's CRT display.}

BEGIN
    WRITELN ('ICOL C',COLOR:1);
    WRITELN ('IVEC ',X:1,' ',Y:1);

    CASE SYMBOL OF
        SA_1: WRITELN ('IRVE 4,-15,-8,0,4,0,0,15,-2,-3,2,3');
        SA_2: WRITELN ('IRVE 5,-15,-9,0,0,3,9,8,-2,3,-6,0,3,0');
        SA_3: WRITELN ('IRVE -5,-15,4,0,4,6,-5,2,7,7,-10,0,5,0');
        SA_4: WRITELN ('IRVE 3,-10,-8,0,5,10,0,-15,0,15');
        SA_5: WRITELN ('IRVE -4,-12,6,0,2,4,-2,4,-6,0,0,5,8,0,-4,0');
        SA_6: WRITELN ('IRVE -5,-8,2,1,6,0,2,-3,-2,-3,-6,0,-2,5,1,4,4,2,0');
        SA_7: WRITELN ('IRVE -1,-14,0,6,1,3,5,5,-10,0,5,0');
        SA_8: WRITELN ('IRVE 0,0,-5,0,0,-12,10,0,0,7,-10,0,10,0,0,5,-5,0');
        SA_9: WRITELN ('IRVE -5,-15,10,0,0,14,-10,0,0,-8,10,0,0,8,-5,0');
        HAWK: WRITELN ('IRVE -5,0,0,-14,0,7,10,0,0,-7,0,14');
        BAR_2: WRITELN ('IRVE 5,-14,-9,0,0,3,5,3,-4,0,8,0,-4,0,4,5,-2,3,-6,0');
        CROT: WRITELN ('IRVE 6,0,-10,0,-2,-3,0,-10,2,-3,10,0');
        WING: WRITELN ('IRPOL 0,0,-8,-8,3,0,3,4,0,3,-3,3,0,-8,8');
        AIR: WRITELN ('IRPOL 0,0,-8,-11,16,0,-8,11,0,-11,-3,-3,6,0,-3,3');
        AAA_0: WRITELN ('IRVE -5,-10,9,0,3,-5,-7,15,-8,-15,8,15');
        AAA_1: BEGIN
            WRITE ('IRVE -5,-10,9,0,3,-5,-7,15,-8,-15,8,15');
            WRITELN ('IRVE 0,-6,-1,-1,2,0,-1,1,IRVE -3,-5');
        END
    END

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMAT:CTHAMESIALR46CGS.PAS:309

```

699      3      END;
700      2      AAA_2: BEGIN
701      3          WRITE ('IRVE -5,-10,9,0,3,-5,-7,15,-8,-15,8,15');
702      3          WRITE ('IRVE 0,-6,-1,-1,2,0,-1,1,IRVE -3,-5');
703      3          WRITELN ('IRVE 0,0,-1,-1,2,0,-1,1,IRVE 6,0');
704      3      END;
705      2      AAA_3: BEGIN
706      3          WRITE ('IRVE -5,-10,9,0,3,-5,-7,15,-8,-15,8,15');
707      3          WRITELN ('IRVE 0,-6,-1,-1,2,0,-1,1,IRVE -3,-5');
708      3          WRITE ('IRVE 0,0,-1,-1,2,0,-1,1,IRVE 6,0');
709      3          WRITELN ('IRVE 0,0,-1,-1,2,0,-1,1,IRVE -3,13');
710      3      END;
711      2      STAF2: WRITELN('IRVE 0,0,0,-20,0,17,IRVE 5,-15,-9,0,0,3,9,8,-2,3,-6,0,3,0');
712      2      STAF6: WRITELN('IRVE 0,0,0,-20,0,17,IRVE -5,-8,2,1,6,0,2,-3,-2,-3,-6,0,-2,5,1,4,4,2,0');

713      2      STAF8: WRITELN('IRVE 0,0,0,-20,0,17,IRVE 0,0,-5,0,0,-12,10,0,0,7,-10,0,10,0,0,5,-5,0');
714      2      UNK0: WRITELN('IRVE -6,0,0,-15,12,0,0,15,IRVE -6,0');
715      2      UNK1: WRITELN('IRVE -6,0,0,-15,12,0,0,15,IRVE -6,0,IRVE 0,-6,-1,-1,2,0,-1,1,IRVE -3,-5');

716      2      UNK2: BEGIN
717      3          WRITE ('IRVE -6,0,0,-15,12,0,0,15,IRVE -6,0,IRVE 0,-6,-1,-1,2,0,-1,1,IRVE -3,-5'
);
718      3          WRITELN('IRVE 0,0,-1,-1,2,0,-1,1,IRVE 6,0')
719      3      END;
720      2      UNK3: BEGIN
721      3          WRITELN('IRVE -6,0,0,-15,12,0,0,15,IRVE -6,0,IRVE 0,-6,-1,-1,2,0,-1,1,IRVE -3,-5'
);
722      3          WRITELN('IRVE 0,0,-1,-1,2,0,-1,1,IRVE 6,0,IRVE 0,0,-1,-1,2,0,-1,1,IRVE -3,13')
723      3      END;
724      2      OTHERWISE WRITELN ('ERROR--PROCEDURE = OUTPUT_A_SYMBOL');
725      2
726      2      END; {END OF CASE}
727      1      END;
728      0
729      0
730      0

```

{END OF OUTPUT_A_SYMBOL}

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	IMA1:[THAMES]ALR46CGS.PAS:309


```

731 0
732 0
733 0
734 0
735 1
736 2
737 2
738 2
739 2
740 2
741 3
742 3
743 3
744 3
745 2
746 2
747 3
748 3
749 3
750 3
751 3
752 3
753 3
754 2
755 2
756 3
757 3
758 3
759 3

760 3
761 3

      {BEGIN DRIVE_PILOTS_DISPLAY}

      WITH CURRENT_THREAT DO
      BEGIN
        IF (RANGE > MAXIMUM_RWR_RANGE) AND (STATUS = NEW) THEN
          {IF THE NEW THREAT IS OUT OF RANGE - THEN}
          STATUS := OUT_OF_RANGE
        ELSE
          {IF AN OLD THREAT IS OUT OF RANGE - THEN}
          IF (RANGE > MAXIMUM_RWR_RANGE) AND (STATUS = UPDATED) THEN
            BEGIN
              OUTPUT_A_SYMBOL (SYMBOL, BACKGROUND_COLOR, OLD_X_POSITION, OLD_Y_POSITION);
              STATUS := OUT_OF_RANGE
            END {END OF IF}
          ELSE
            {IF THE NEW THREAT IS IN RANGE - THEN}
            IF (RANGE < MAXIMUM_RWR_RANGE) AND (STATUS = NEW) THEN
              BEGIN
                STATUS := UPDATED;
                OUTPUT_A_SYMBOL (SYMBOL, COLOR, NEW_X_POSITION, NEW_Y_POSITION);
                {REMEMBER WHERE THE SYMBOL IS ON THE DISPLAY}
                OLD_X_POSITION := NEW_X_POSITION;
                OLD_Y_POSITION := NEW_Y_POSITION;
              END {END OF ELSE}
            ELSE
              {IF OLD THREAT IS STILL IN RANGE - THEN}
              IF (RANGE < MAXIMUM_RWR_RANGE) AND (STATUS = UPDATED) THEN
                BEGIN
                  {REMOVE THREAT FROM OLD POSITION}
                  OUTPUT_A_SYMBOL (SYMBOL, BACKGROUND_COLOR, OLD_X_POSITION, OLD_Y_POSITION);
                  {OUTPUT SYMBOL TO NEW POSITION}
                  OUTPUT_A_SYMBOL (SYMBOL, COLOR, NEW_X_POSITION, NEW_Y_POSITION);
                  {REMEMBER WHERE THE SYMBOL IS ON THE DISPLAY}
                  OLD_X_POSITION := NEW_X_POSITION;
                  OLD_Y_POSITION := NEW_Y_POSITION
                END
              END
            END
          END
        END
      END
    
```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:[THAMES]ALR46CGS.PAS:309

```

762      3      END (END OF IF)
763      3      ELSE
764      2      IF (RANGE < MAXIMUM_RWR_RANGE) AND (STATUS = REMOVE) THEN
765      2      (REMOVE THE SYMBOL FROM THE DISPLAY)
766      2      OUTPUT_A_SYMBOL (SYMBOL, BACKGROUND_COLOR, OLD_X_POSITION, OLD_Y_POSITION)
767      2      END; (END OF WITH)
768      1      END; (DRIVE_PILOTS_DISPLAY)
769      0
770      0
771      0
772      0
773      4      (PROCEDURE SEND_CRITICAL_PARAMETERS)
774      4      PROCEDURE SEND_CRITICAL_PARAMETERS (VAR CURRENT_THREAT ; RWR_DISPLAY_DATA;
775      4      VAR EMITTER_NUMBER ; INTEGER);
776      4
777      4      {This procedure sends the emitter number, symbol, range, and azimuth to the
778      4      4027 CRT for each active threat.}
779      4      BEGIN
780      4      (SEND_CRITICAL_PARAMETERS)
781      0      (OUTPUT THE EMITTER NUMBER AT THE CORRECT LOCATION)
782      1      WRITELN ('!VEC 0',(336 - (EMITTER_NUMBER * 14));1);
783      1      WITH CURRENT_THREAT DO
784      2      BEGIN
785      2      IF STATUS = REMOVE THEN
786      2      (REMOVE THE TREAT'S CRITICAL PARAMETER DATA FROM THE DISPLAY)
787      2      WRITELN ('!STR/'
788      2      ELSE
789      2      (DISPLAY THE CRITICAL PARAMETERS)
790      2      WRITELN ('!STR/'EMITTER_NUMBER:3;SYMBOL:10;ROUND(RANGE):6;ROUND(AZIMUTH):7;');
791      1      END; (END OF WITH)
792      0      (END OF SEND_CRITICAL_PARAMETERS)
793      0

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DM1:CTHAMESJALR46CGS.PAS;309 .


```

794      FUNCTION X_COORDINATE (VAR MAXIMUM_RWR_RANGE : REAL;           {FUNCTION X_COORDINATE}
795      VAR RANGE : REAL;
796      VAR AZIMUTH : REAL) : INTEGER;
797
798      {This function generates the x.coordinate needed to position the threat symbol
799      on the 4027 CRT display. The value is generated based on the threat's range
800      and azimuth.}
801
802      CONST
803          PI = 3.1415926;
804
805      VAR
806          RADIAN_ANGLE : REAL;
807          RADIAN_ANGLE := 360.0 / (2 * PI);
808          RADIANS_TO_DEGREES := (2 * PI) / 360.0;
809          X_SCALE_FACTOR := RANGE * SIN (AZIMUTH * DEGREES_TO_RADIANS) / MAXIMUM_RWR_RANGE;
810          X_COORDINATE := 425 - ROUND (170.0 * X_SCALE_FACTOR); {RETURN THE 4027'S X COORDINATE}
811
812      BEGIN
813          X_COORDINATE := 425 - ROUND (170.0 * X_SCALE_FACTOR); {END OF X_COORDINATE}
814
815      END;
816
817      FUNCTION Y_COORDINATE (VAR MAXIMUM_RWR_RANGE : REAL;           {FUNCTION Y_COORDINATE}
818      VAR RANGE : REAL;
819      VAR AZIMUTH : REAL) : INTEGER;
820
821      {This function generates the y.coordinate needed to position the threat symbol
822      on the 4027 CRT display. The value is generated based on the threat's range
823      and azimuth.}
824
825

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DNA1:ETHAMESJALR46CGS.PAS;309


```

826 4
827 4
828 4
829 4
830 4
831 4
832 4
833 4
834 4
835 4
836 0
837 1
838 1
839 1
840 1
841 0
842 0
843 0
844 0
845 0
846 0
847 0
848 0
849 1
850 2
851 2
852 3
853 3
854 4
855 4
856 4
857 4

CONST
  PI = 3.1415926;
VAR
  RADIANS_ANGLE : REAL;
  RADIANS_TO_DEGREES : REAL;
  DEGREES_TO_RADIANS : REAL;
  Y_SCALE_FACTOR : REAL;

  BEGIN
    RADIANS_TO_DEGREES := 360.0 / (2 * PI);
    DEGREES_TO_RADIANS := (2 * PI) / 360.0;
    Y_SCALE_FACTOR := RANGE * COS (AZIMUTH * DEGREES_TO_RADIANS) / MAXIMUM_RWR_RANGE;
    Y_COORDINATE := 177 + ROUND (170.0 * Y_SCALE_FACTOR); {RETURN THE 4027'S Y COORDINATE}
  END; {Y_COORDINATE}

  {BEGIN Y_COORDINATE}

  BEGIN
    {BEGIN OUTPUT_TO_RWR_DISPLAY}

    {DELETE EMITTERS THAT HAVE SHUT DOWN}

    IF NON_ACTIVE_EMITTERS < 1 THEN {IF THERE ARE EMITTERS THAT HAVE TO BE DELETED - THEN}
      BEGIN
        FOR EMITTER_NUMBER := 1 TO 16 DO
          BEGIN
            IF EMITTER_NUMBER IN NON_ACTIVE_EMITTERS THEN
              BEGIN
                {IF THIS EMITTER HAS TO BE DELETED - THEN}
                CURRENT_THREAT := EMITTER_POSITIONS [EMITTER_NUMBER];
                CURRENT_THREAT_COLOR := BACKGROUND_COLOR;
                {GET THE THREAT'S DISPLAY DATA}
                {IF THE THREAT IS OUT OF RANGE-THEN}
                IF CURRENT_THREAT_STATUS = OUT_OF_RANGE THEN

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:ETHAMESJALR46CGS.PAS;309


```

858      CURRENT_THREAT.STATUS := REMOVE
859      ELSE
860      BEGIN
861          CURRENT_THREAT.STATUS := REMOVE;
862          DRIVE_PILOTS_DISPLAY (CURRENT_THREAT, MAXIMUM_FWR_RANGE, BACKGROUND_COLOR)
863      END;
864
865      {REFRESH THE DISPLAY'S CRITICAL PARAMETER DATA}
866      SEND_CRITICAL_PARAMETERS(CURRENT_THREAT, EMITTER_NUMBER);
867      EMITTER_POSITIONS [EMITTER_NUMBER] := CURRENT_THREAT;
868      END; {END OF IF}
869      END; {END OF FOR}
870      NON_ACTIVE_EMITTERS := [];
871      END; {END OF IF}
872
873      {PROCESS NEW OR UPDATED EMITTERS}
874
875      IF MODIFIED_EMITTERS <> [] THEN {IF THERE ARE THREAT PARAMETERS THAT WERE MODIFIED - THEN}
876      BEGIN
877          CURRENT_EMITTER := FIRST_EMITTER;
878          FOR EMITTER_NUMBER := 1 TO 16 DO
879              BEGIN
880                  {IF THIS THREAT'S PARAMETERS WERE MODIFIED - THEN}
881                  IF EMITTER_NUMBER IN MODIFIED_EMITTERS THEN
882                      WITH CURRENT_EMITTER DO
883                          BEGIN
884                              IF (EMITTER_NUMBER IN MODIFIED_EMITTERS) AND
885                                  ((1 IN UPDATED_ENTRIES) OR (17 IN UPDATED_ENTRIES)) THEN
886                                  BEGIN
887                                      CURRENT_THREAT := EMITTER_POSITIONS [EMITTER_NUMBER];
888                                      CURRENT_THREAT.SYMBOL := SYMBOL;
889                                      CURRENT_THREAT.RANGE := RANGE;
890                                      CURRENT_THREAT.AZIMUTH := AZIMUTH;

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:ITHAMESIALR46CGS.FAS:309


```

890      CURRENT_THREAT.NEW_X_POSITION :=
891          X_COORDINATE (MAXIMUM_RWR_RANGE,RANGE,AZIMUTH);
892      CURRENT_THREAT.NEW_Y_POSITION :=
893          Y_COORDINATE (MAXIMUM_RWR_RANGE,RANGE,AZIMUTH);
894      IF (CURRENT_THREAT.STATUS = REMOVE) THEN {IF NEW THREAT THEN}
895          BEGIN
896              GET_NEXT_COLOR (CURRENT_THREAT.SYMBOL,CURRENT_THREAT.COLOR,
897                  CURRENT_COLOR,BACKGROUND_COLOR);
898              CURRENT_THREAT.STATUS := NEW
899          END
900      ELSE
901          IF (CURRENT_THREAT.STATUS = OUT_OF_RANGE) AND
902              (RANGE <= MAXIMUM_RWR_RANGE) THEN
903              CURRENT_THREAT.STATUS := NEW; {IT LOOKS LIKE A NEW THREAT}
904      IF CURRENT_THREAT.STATUS <> OUT_OF_RANGE THEN
905          DRIVE_PLOTS_DISPLAY (CURRENT_THREAT,MAXIMUM_RWR_RANGE,
906              BACKGROUND_COLOR);
907      EMITTER_POSITIONS [EMITTER_NUMBER] := CURRENT_THREAT;
908      MODIFIED_EMITTERS := MODIFIED_EMITTERS - [EMITTER_NUMBER];
909      SEND_CRITICAL_PARAMETERS(CURRENT_THREAT,EMITTER_NUMBER);
910      END; {END OF IF}
911      END; {END OF WITH}
912      CURRENT_EMITTER := CURRENT_EMITTER^.NEXT_EMITTER; {GET THE NEXT_EMITTER}
913      END; {END OF FOR}
914      MODIFIED_EMITTERS := []; {SHOW THAT WE HAVE PROCESSED ALL MODIFIED_EMITTERS}
915      END; {END OF IF}
916      END;
917
918      0
919      0
920      3
921      3

```

```

PROCEDURE INITIALIZE_POSITION (VAR EMITTER_POSITIONS : TYPE_LOCATION);
{PROCEDURE INITIALIZE_POSITION}

```

```

9922 {This procedure initializes the records used to maintain the position of the
9923 symbols on the simulated pilot's display.}
9924
9925 VAR
9926     CURRENT_THREAT : RWR_DISPLAY_DATA;
9927     EMITTER_NUMBER : INTEGER;
9928 BEGIN
9929     WITH CURRENT_THREAT DO
9930         BEGIN
9931             STATUS := REMOVE;
9932             SYMBOL := UNKO;
9933             RANGE := 49.0;
9934             AZIMUTH := 180.0;
9935             NEW_X_POSITION := 80;
9936             NEW_Y_POSITION := 100;
9937             OLD_X_POSITION := 40;
9938             OLD_Y_POSITION := 50;
9939             COLOR := 7
9940         END; {END OF WITH}
9941         FOR EMITTER_NUMBER := 1 TO 16 DO
9942             EMITTER_POSITIONS [EMITTER_NUMBER] := CURRENT_THREAT;
9943         END;
9944     END;
9945
9946
9947 BEGIN
9948     BACKGROUND_COLOR := 2;
9949     CURRENT_COLOR := 0;
9950     MAXIMUM_RWR_RANGE := 200.0;
9951     READ_SYMBOL_TYPE_FILE (SYMBOL_TYPE);
9952     INITIALIZE_POSITION (EMITTER_POSITIONS);
9953     CREATE_TRACK_FILE_ENTRIES (FIRST_EMITTER, CURRENT_EMITTER);

```

ALR46CGS		10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
		11-NOV-1981	19:45:54	DMA1:ETHAMESIALR46CGS.PAS;309
954	1			SET STATE_OF_4027(BACKGROUND_COLOR);
955	1			OPEN (ALR46_DATA,'(ETHAMESIRWKDATA.',OLD);
956	1			RESET (ALR46_DATA);
957	1			REPEAT
958	2			READLN (ALR46_DATA,MONITOR_DATA1);
959	2			READLN (ALR46_DATA,MONITOR_DATA2);
960	2			READLN (ALR46_DATA,MONITOR_DATA3);
961	2			READLN (ALR46_DATA,_MONITOR_DATA4);
962	2			ALR46_DATA_INDEX := 1;
963	2			REPEAT
964	3			IF MONITOR_DATA [ALR46_DATA_INDEX + 1] = PC_FLAG THEN
965	3			ALR46_DATA_INDEX := ALR46_DATA_INDEX + 4
966	3			ELSE
967	3			BEGIN
968	4			UPDATE_TRACK_FILE_ENTRY (MONITOR_DATA,CURRENT_EMITTER,FIRST_EMITTER,NON_ACTIVE_EMITTERS,
969	4			MODIFIED_EMITTERS,OLD_EMITTERS,ACTIVE_EMITTERS,ADDRESS_STATUS,EMITTER_PRIORITY,
970	4			IMAGE_OF_ALR46_TRACK_FILE,ALR46_FREE_POINTER,ALR46_USED_POINTER,
971	4			START_ADDRESS_OF_TRACK_FILE,ALR46_DATA_INDEX,SYMBOL_TYPE);
972	4			ALR46_DATA_INDEX := ALR46_DATA_INDEX + 2;
973	4			OUTPUT_TO_RWR_DISPLAY (FIRST_EMITTER,MODIFIED_EMITTERS,NON_ACTIVE_EMITTERS,
974	4			EMITTER_POSITIONS,BACKGROUND_COLOR,MAXIMUM_RWR_RANGE,
975	4			CURRENT_COLOR);
976	4			END;
977	3			UNTIL ALR46_DATA_INDEX >= 4;
978	2			UNTIL EOF (ALR46_DATA);
979	1			CLOSE (ALR46_DATA)
980	1			END;
981	0			
982	0			
983	0			
984	0			
985	2			PROCEDURE PARAMETER (VAR MONITOR_DATA : ARRAY[4000];
				{PROCEDURE PARAMETER}

10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
11-NOV-1981	19:45:54	DMA1:CTHAMESJALK46CGS.PAS;309

ALR46CGS	986	2	VAR CURRENT_EMITTER : EMITTER_PTR;
	987	2	VAR FIRST_EMITTER : EMITTER_PTR;
	988	2	VAR NON_ACTIVE_EMITTERS : TYPE_EMITTER;
	989	2	VAR MODIFIED_EMITTERS : TYPE_EMITTER;
	990	2	VAR OLD_EMITTERS : TYPE_EMITTER;
	991	2	VAR ACTIVE_EMITTERS : TYPE_EMITTER;
	992	2	VAR ADDRESS_STATUS : ADDRESS;
	993	2	VAR EMITTER_PRIORITY : ARRAY17;
	994	2	VAR IMAGE_OF_ALR46_TRACK_FILE : ALR46ARRAY;
	995	2	VAR ALR46_E_POINTER : INTEGER;
	996	2	VAR ALR46_LL_POINTER : INTEGER;
	997	2	VAR START_ADDRESS_OF_TRACK_FILE : INTEGER;
	998	2	
	999		
	1000		{This procedure is the executive for the parameter display process. It must
	1001		set the VT-100 into the proper state required by the parameter mode, decode
	1002	2	the words sent by the hardware monitor, and control the critical parameter
	1003	2	display.}
	1004	2	CONST
	1005	2	PC_FLAG = %0177777;
	1006	2	VAR
	1007	2	ESC : CHAR;
	1008	2	ALR46_DATA_INDEX : INTEGER;
	1009	2	SYMBOL_TYPE : SYMBOL_TYPE_STRUCTURE;
	1010	2	
	1011	2	
	1012	3	PROCEDURE POSITION_CURSOR(HORIZONTAL,VERTICAL : INTEGER); {PROCEDURE POSITION_CURSOR}
	1013	3	
	1014	3	
	1015		{This procedure positions the cursor to the correct row and column on the
	1016		VT-100. The position is based on the specific critical parameter being
	1017	3	updated.}

10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29	
11-NOV-1981	19:45:54	IMA1:ETHAMESJALR46CGS,FAS:309	

1018	3	BEGIN	{BEGIN POSITION_CURSOR}
1019	3	ESC := CHR(27);	
1020	0	IF HORIZONTAL >= 22 THEN	{IF WE ARE IN THE OPERATOR INTERFACE REGION THEN}
1021	1	WRITE (ESC,'(B',	{SET ASCII MODE}
1022	1	ESC,'(O',	{SET NO UNDERSCORE MODE}
1023	1	ESC,'(O',	{POSITION_CURSOR}
1024	1	ESC,'(',HORIZONTAL:1,';',VERTICAL:1,'')	
1025	1	ELSE	
1026	1	WRITE (ESC,'(O',	{SET GRAPHICS MODE}
1027	1	ESC,'(A',	{SET UNDERSCORE MODE}
1028	1	ESC,'(',HORIZONTAL:1,';',VERTICAL:1,'')	{POSITION_CURSOR}
1029	1	END;	{END OF POSITION_CURSOR}
1030	0		
1031	0		
1032	0		
1033	3	PROCEDURE SET_STATE_OF_VT100;	{PROCEDURE SET_STATE_OF_VT100}
1034	3		
1035		{This procedure sets the VT-100 into the state required by the critical	
1036		parameter process. For example, the VT-100 must be set into the one hundred	
1037		and thirty two column and twenty four line state during the execution of the	
1038	3	parameter mode.}	
1039	3	TYPE	
1040	3	NAME = PACKED ARRAY [1..15] OF CHAR;	
1041	3		
1042	3	PROCEDURE GENERATE_FORM(NAME_OF_VARIABLE : NAME);	{PROCEDURE GENERATE_FORM}
1043	4		
1044	4		
1045		{This function generates the blank form on the VT-100 that will be filled in	
1046	4	later by critical parameters.}	
1047	4		
1048	4	VAR	
1049	4	VERTICAL_LINE_COUNT : INTEGER;	

AD-A119 252

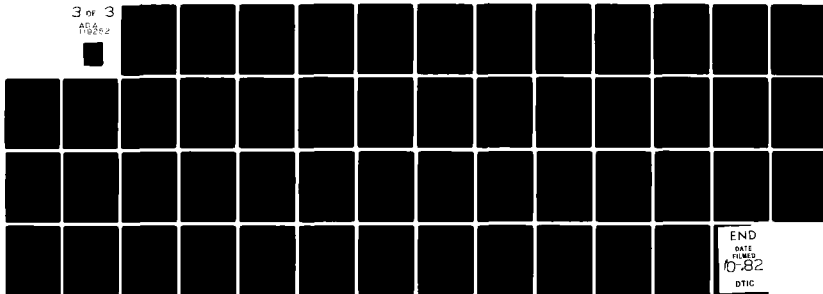
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCH00--ETC F/G 9/2
ALR-46 COMPUTER GRAPHICS SYSTEM FOR THE ROBINS AFB ELECTRONIC W--ETC(U)
DEC 81 J W THAMES
AFIT/GCS/EE/81D-18

UNCLASSIFIED

NL

3 of 3

ALL
PAGE 2



END
DATE
FILMED
10-82
DTIC

ALR46CGS 10-DEC-1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
 11-NOV-1981 19:45:54 DNAME:ITHAMESTALR46CGS.PAS:309

```

1050 4 BEGIN
1051 0 ESC := CHR(27);
1052 1 WRITE(NAME_OF_VARIABLE:15);
1053 1 FOR VERTICAL_LINE_COUNT := 1 TO 15 DO
1054 1 WRITE('x ');
1055 1 WRITELN('x x');
1056 1 END;
1057 0
1058 0
1059 0
1060 0 BEGIN
1061 0 ESC := CHR(27);
1062 1 WRITELN(ESC,'[?3N');
1063 1 WRITELN(ESC,'<ESC,'#8');
1064 1 WRITELN(ESC,'[22;24r');
1065 1 WRITELN(ESC,'[2J');
1066 1 POSITION_CURSOR(0,0);
1067 1
1068 1
1069 1
1070 1 GENERATE_FORM('BY PRIORITY ');
1071 1 GENERATE_FORM('SYMBOL ');
1072 1 GENERATE_FORM('TYPE ');
1073 1 GENERATE_FORM('BANDS ');
1074 1 GENERATE_FORM('PRI/FRAME PER ');
1075 1 GENERATE_FORM('PULSE TRAIN DES');
1076 1 GENERATE_FORM('PRI AVERAGE ');
1077 1 GENERATE_FORM('DEVIATION ');
1078 1 GENERATE_FORM('SPECIAL ');
1079 1 GENERATE_FORM('POWER/PWR ROUNO');
1080 1 GENERATE_FORM('HIGH BAND SINE ');
1081 1 GENERATE_FORM('HIGH BAND COS ');

```

{BEGIN GENERATE_FORM}

{END OF GENERATE_FORM}

{BEGIN SET_STATE_OF_VT100}

{ SET CRT IN 132 COLUMN MODE }

{ SET CRT IN 24 LINE X 132 COLUMN MODE }

{ SET SCROLL REGION BETWEEN LINES 22 & 24 }

{ ERASE SCREEN }

{POSITION CURSOR TO TOP OF SCREEN}

{GENERATE A BLANK CRITICAL PARAMETER FORM}

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:[THAMES]ALR46CGS.PAS:309


```

1082      1      GENERATE_FORM ('CEPC      ');
1083      1      GENERATE_FORM ('HI BAND AGE CNT');
1084      1      GENERATE_FORM ('CD BAND AGE CNT');
1085      1      GENERATE_FORM ('PRIORITY  ');
1086      1      GENERATE_FORM ('PRIORITY /ROUND');
1087      1      GENERATE_FORM ('AZIMUTH   ');
1088      1      GENERATE_FORM ('LETHALITY RING ');
1089      1      GENERATE_FORM ('NL AGE    ');
1090      1      GENERATE_FORM ('RECORD NUMBER ');
1091      1      POSITION_CURSOR (24,0);
1092      1      WRITE  ('ALR46 CGS');
1093      1      END;
1094      0
1095      0
1096      0
1097      0
1098      3      {PROCEDURE OUTPUT_CRITICAL_PARAMETER (VAR FIRST_EMITTER,CURRENT_EMITTER :EMITTER_PTR;
1099      3      VAR MODIFIED_EMITTERS : TYPE_EMITTER;
1100      3      VAR NON_ACTIVE_EMITTERS : TYPE_EMITTER;
1101      3      VAR ADDRESS_STATUS :ADDRESS;
1102      3      VAR EMITTER_PRIORITY : ARRAY17);
1103      3
1104
1105      {This procedure controls and displays the critical parameters on the VT-100
1106      3      CRT. Each parameter changed is written to the display. When a threat stops
1107      3      transmitting, its parameters are removed from the display.}
1108      3      VAR
1109      3      EMITTER_NUMBER : INTEGER;
1110      3      ENTRY_NUMBER : INTEGER;
1111      3      BEGIN
1112      0      ESC := CHR(27);
1113      1      IF ADDRESS_STATUS = COMPLETED THEN {IF THE PRIORITY OF A THREAT HAS BEEN CHANGED - THEN}

```

{POSITION THE CURSOR TO LINE 24}

{END OF SET_STATE_OF_VT100}

{PROCEDURE OUTPUT_CRITICAL_PARAMETER}

VAR FIRST_EMITTER,CURRENT_EMITTER :EMITTER_PTR;

VAR MODIFIED_EMITTERS : TYPE_EMITTER;

VAR NON_ACTIVE_EMITTERS : TYPE_EMITTER;

VAR ADDRESS_STATUS :ADDRESS;

VAR EMITTER_PRIORITY : ARRAY17;

{This procedure controls and displays the critical parameters on the VT-100 CRT. Each parameter changed is written to the display. When a threat stops transmitting, its parameters are removed from the display.}

VAR

EMITTER_NUMBER : INTEGER;

ENTRY_NUMBER : INTEGER;

BEGIN

ESC := CHR(27);

IF ADDRESS_STATUS = COMPLETED THEN {IF THE PRIORITY OF A THREAT HAS BEEN CHANGED - THEN}

ALR44CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:ITHAMESJALR44CGS.PAS;309

```

1114      1      BEGIN
1115      2      FOR EMITTER_NUMBER := 1 TO EMITTER_PRIORITY[1] DO      {OUTPUT THE EMITTER PRIORITY}
1116      2      BEGIN
1117      3      POSITION_CURSOR (1,(EMITTER_NUMBER - 1)*7 + 17);
1118      3      Writeln (EMITTER_PRIORITY[EMITTER_NUMBER+1];3);
1119      3      ADDRESS_STATUS := NONE_IN_PROGRESS;      {EMITTER PRIORITY HAS BEEN OUTPUT}
1120      3      POSITION_CURSOR (24,0)      {PLACE CURSOR IN OPERATOR REGION}
1121      3      END; {END OF FOR}
1122      2      END; {END OF IF}
1123      1      IF NON_ACTIVE_EMITTERS <> [] THEN      {IF ANY THREATS HAVE STOPPED TRANSMITTING - THEN}
1124      1      BEGIN
1125      2      FOR EMITTER_NUMBER := 1 TO 16 DO
1126      2      BEGIN
1127      3      IF EMITTER_NUMBER IN NON_ACTIVE_EMITTERS THEN
1128      3      BEGIN
1129      4      FOR ENTRY_NUMBER := 0 TO 20 DO
1130      4      BEGIN
1131      5      POSITION_CURSOR (ENTRY_NUMBER+1,(EMITTER_NUMBER-1)*7 + 17);
1132      5      Writeln(' ')      {CLEAR THIS EMITTER ENTRY}
1133      5      END; {END OF FOR}
1134      4      END; {END OF IF}
1135      3      END; {END OF FOR}
1136      2      POSITION_CURSOR (24,0);      {PLACE CURSOR IN OPERATOR REGION}
1137      2      NON_ACTIVE_EMITTERS := [];      {CALL EMITTERS NO LONGER UP, HAVE BEEN CLEARED}
1138      2      END; {END OF IF}
1139      1      {IF THE CRITICAL PARAMETERS OF ANY THREAT HAVE BEEN CHANGED - THEN}
1140      1      IF MODIFIED_EMITTERS <> [] THEN
1141      1      BEGIN
1142      2      CURRENT_EMITTER := FIRST_EMITTER;
1143      2      FOR EMITTER_NUMBER := 1 TO 16 DO
1144      2      BEGIN
1145      3      {IF THIS THREAT'S CRITICAL PARAMETERS HAVE BEEN CHANGED - THEN}

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DNA1:CTHAMESJALR46CGS.PAS;309

```

1146      3      IF EMITTER_NUMBER IN MODIFIED_EMITTERS THEN
1147      3      BEGIN
1148      4      WITH CURRENT_EMITTER DO
1149      4      FOR ENTRY_NUMBER := 1 TO 20 DO
1150      4      BEGIN
1151      5      (IF THIS CRITICAL PARAMETER HAVE BEEN CHANGED - THEN)
1152      5      IF ENTRY_NUMBER IN UPDATED_ENTRIES THEN
1153      5      BEGIN
1154      6      POSITION_CURSOR (ENTRY_NUMBER+1,(EMITTER_NUMBER-1)*7 + 17);
1155      6      CASE ENTRY_NUMBER OF
1156      7      1 :WRITELN(SYMBOL:5);
1157      7      2 :WRITELN('T',EMITTER_TYPE:4);
1158      7      3 :WRITELN(RF_BAND:4);
1159      7      4 :WRITELN(PRI_FRAME_PERIOD:4:1);
1160      7      5 :BEGIN
1161      8      IF (PULSE_TRAIN_DESCR = JIT) OR
1162      8      (PULSE_TRAIN_DESCR = STAG) THEN
1163      8      WRITELN(PULSE_TRAIN_DESCR:6)
1164      8      ELSE
1165      8      IF PULSE_TRAIN_DESCR = STAG THEN
1166      8      WRITELN (STAGGER_LEVEL:1,PULSE_TRAIN_DESCR:5);
1167      8      END; {END 5}
1168      7      6 :WRITELN(PRI_AVERAGE:5:1);
1169      7      7 :WRITELN(DEVIATION:5:1);
1170      7      8 :WRITELN(SPECIAL_FIELD:4);
1171      7      9 :WRITELN(POWER_PRROUND:4);
1172      7      10:WRITELN(HI_BAND_SIN:4);
1173      7      11:WRITELN(HI_BAND_COS:4);
1174      7      12:WRITELN(CEPC:4);
1175      7      13:WRITELN(HI_BAND_AGE_CNT:4);
1176      7      14:WRITELN(CO_AGE_COUNTER:4);
1177      7      15:WRITELN(PRIORITY:4);

```

ALR46CGS	10-DEC-1981	18:08:39	WAX-11 PASCAL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:[THAMES]ALR46CGS.PAS;309


```

1178      7      16:WRITELN(PRIO_POWERDOWN:4);
1179      7      17:WRITELN(TRUNC(AZIMUTH):4);
1180      7      18:WRITELN(LETHALITY_RING:4:1);
1181      7      19:WRITELN(ML_AGE:4);
1182      7      20:WRITELN(RECORD_NUMBER:4)
1183      7      OTHERWISE WRITELN('ERROR--PROCEDURE = OUTPUT_CRITICAL_PARAMETER')
1184      7      END; {END OF CASE}
1185      6      END; {END OF IF}
1186      5      END; {END OF FOR}
1187      4      END; {END OF IF}
1188      3      CURRENT_EMITTER:=UPDATED_ENTRIES := []; {ALL MODIFIED ENTRIES HAVE BEEN DISPLAYED}
1189      3      CURRENT_EMITTER := CURRENT_EMITTER^.NEXT_EMITTER; {GET READY FOR NEXT EMITTER}
1190      3      END; {END OF FOR}
1191      2      POSITION_CURSOR (24,0);
1192      2      MODIFIED_EMITTERS := [];
1193      2      END; {END OF IF}
1194      1      END;
1195      0
1196      0
1197      0
1198      0
1199      0
1200      0      BEGIN
1201      0      CREATE_TRACK_FILE_ENTRIES (FIRST_EMITTER,CURRENT_EMITTER);
1202      1      READ_SYMBOL_TYPE_FILE (SYMBOL_TYPE);
1203      1      SET_STATE_OF_VT100;
1204      1      OPEN (ALR46_DATA,'[THAMES]PARAMDATA','OLD');
1205      1      RESET (ALR46_DATA);
1206      1      REPEAT
1207      2      READLN (ALR46_DATA,MONITOR_DATA[1]);
1208      2      READLN (ALR46_DATA,MONITOR_DATA[2]);
1209      2      READLN (ALR46_DATA,MONITOR_DATA[3]);

```

ALR46CGS	10-DEC-1981	18:08:39	VAX-11 PA: AL VERSION V1.1-29
	11-NOV-1981	19:45:54	DMA1:CTHA:ESJALR46CGS.PAS:309


```

1210 READLN (ALR46_DATA,MONITOR_DATA[4]);
1211 ALR46_DATA_INDEX := 1;
1212 REPEAT
1213 3
1214 3 IF MONITOR_DATA [ALR46_DATA_INDEX + 1] = PC_FLAG THEN
1215 3 ALR46_DATA_INDEX := ALR46_DATA_INDEX + 4
1216 3 ELSE
1217 3 BEGIN
1218 4 UPDATE_TRACK_FILE_ENTRY (MONITOR_DATA,CURRENT_EMITTER,FIRST_EMITTER,NON_ACTIVE_EMITTERS,
1219 4 MODIFIED_EMITTERS,OLD_EMITTERS,ACTIVE_EMITTERS,ADDRESS_STATUS,
1220 4 EMITTER_PRIORITY,IMAGE_OF_ALR46_TRACK_FILE,ALR46_FREE_POINTER,
1221 4 ALR46_USED_POINTER,START_ADDRESS_OF_TRACK_FILE,ALR46_DATA_INDEX,
1222 4 SYMBOL_TYPE);
1223 4 ALR46_DATA_INDEX := ALR46_DATA_INDEX + 2;
1224 4 OUTPUT_CRITICAL_PARAMETER (FIRST_EMITTER,CURRENT_EMITTER,MODIFIED_EMITTERS,
1225 4 NON_ACTIVE_EMITTERS,ADDRESS_STATUS,EMITTER_PRIORITY);
1226 4
1227 3 UNTIL ALR46_DATA_INDEX >= 4;
1228 2 UNTIL EOF (ALR46_DATA);
1229 1 CLOSE (ALR46_DATA)
1230 1 END;
1231 0
1232 0
1233 0 BEGIN
1234 0 REPEAT
1235 2 WRITE ('ALR46 CGS');
1236 2 READ (COMMAND);
1237 2 IF COMMAND IN LIST_OF_LEGAL_COMMANDS THEN
1238 2 BEGIN
1239 3 CASE COMMAND OF
1240 4 HELP : HELPUSER;
1241 4 CRITICAL : PARAMETER (MONITOR_DATA,CURRENT_EMITTER,FIRST_EMITTER,NON_ACTIVE_EMITTERS,

```

ALR46CBS 10-DEC-1981 18:08:39 VAX-11 PASCAL VERSION V1.1-29
 11-NOV-1981 19:45:54 DMA1:CTHAMESJALR46CBS.PAS:309

```

1242      4      MODIFIED_EMITTERS,OLD_EMITTERS,ACTIVE_EMITTERS,ADDRESS_STATUS,
1243      4      EMITTER_PRIORITY,IMAGE_OF_ALR46_TRACK_FILE,ALR46_FREE_POINTER,
1244      4      ALR46_USED_POINTER,START_ADDRESS_OF_TRACK_FILE);
1245      4      RMR : RMR_DISPLAY(MONITOR_DATA,CURRENT_EMITTER,FIRST_EMITTER,NON_ACTIVE_EMITTERS;
1246      4      MODIFIED_EMITTERS,OLD_EMITTERS,ACTIVE_EMITTERS,ADDRESS_STATUS,
1247      4      EMITTER_PRIORITY,IMAGE_OF_ALR46_TRACK_FILE,ALR46_FREE_POINTER,
1248      4      ALR46_USED_POINTER,START_ADDRESS_OF_TRACK_FILE);
1249      4      TERMINATE : STOP_COMMAND := TRUE
1250      4      END; {END OF CASE}
1251      3      END; { IF }
1252      2      UNTIL STOP_COMMAND
1253      2      END.
1254      0

```

(END OF ALR46CBS)

Compilation time = 27.23 seconds (2763 lines/minute).

Active options at end of compilation:
 NODEBUG,NOSTANDARD,LIST,NOCHECK,NOWARNINGS,NOCROSS_REFERENCE,
 NOWMACHINE_CODE,OBJECT,ERROR_LIMIT = 30

A P P E N D I X F

C G S D a t a D i c t i o n a r y

This Appendix contains a description of all of the constants, variables, and processes used in CGS. First, the symbols used in the data dictionary are defined followed by the data dictionary for CGS. The terms described in this data dictionary are the terms used in the actual CGS software.

Data Element Definitions

DATA ELEMENT NAME: Active_emitters

DESCRIPTION: The list of currently active emitters is contained
in Active_emitters.

ALIASES:

DATA ELEMENT NAME: AAA_0

DESCRIPTION: The CGS name for an anti-aircraft artillery (aaa)
threat transmitting in band zero.

ALIASES:

DATA ELEMENT NAME: AAA_1

DESCRIPTION: The CGS name for an anti-aircraft artillery (aaa)
threat transmitting in band one.

ALIASES:

DATA ELEMENT NAME: AAA_2

DESCRIPTION: The CGS name for an anti-aircraft artillery (aaa)
threat transmitting in band two.

ALIASES:

DATA ELEMENT NAME: AAA_3

DESCRIPTION: The CGS name for an anti-aircraft artillery (aaa)
threat transmitting in band three.

ALIASES:

DATA ELEMENT NAME: Address_status

DESCRIPTION: Used by CGS to maintain the status of pointer changes in the ALR-46 OFP. Address_status is initialized to "none_in_progress" when CGS is activated. When the ALR-46 changes ti track_file_free_pointer or word zero of any of CGS's track file entries, address_status is set to in_progress. When the ALR-46 OFP completes the pointer change process, CGS sets address_status back to none_in_progress.

ALIASES:

COMPOSITION: Address_status =

In_progress
Completed
None_in_progress

DATA ELEMENT NAME: AIR

DESCRIPTION: The CGS name for a class of airborne threats.

ALIASES:

DATA ELEMENT NAME: ALR46_free_pointer

DESCRIPTION: Contains the physical address of the first free record in the ALR-46's emitter track file.

ALIASES:

DATA ELEMENT NAME: ALR46_used_pointer

DESCRIPTION: Contains the physical address of the first used record in the ALR-46's emitter track file.

ALIASES:

DATA ELEMENT NAME: Back_ground_color

DESCRIPTION: Represents one of the possible eight colors (0-7) available to the user of the Tektronix 4027 CRT. It is always equal to the number of the background color used while simulating the pilot's RWR display.

ALIASES:

DATA ELEMENT NAME: Bar_2

DESCRIPTION: CGS's name for a special class of surface to air missiles (SAM) called SAM 2 bar.

ALIASES:

DATA ELEMENT NAME: Bits_0_3

DESCRIPTION: As described in appendix A, specific bits of a word in the emitter track file affect different parameters. Specific bits must be extracted from the track file word before they can be examined. Bits_0_3 is used to store bits zero through three of a given word.

ALIASES:

DATA ELEMENT NAME: Bits_6_12

DESCRIPTION: As described in appendix A, specific bits of a word in the emitter track file affect different parameters. Specific bits must be extracted from the track file word before they can be examined. Bits_6_12 is used to store bits six through twelve of a given word.

ALIASES:

DATA ELEMENT NAME: Bits_8_10

DESCRIPTION: As described in appendix A, specific bits of a word in the emitter track file affect different parameters. Specific bits must be extracted from the track file word before they can be examined. Bits_8_10 is used to store bits eight through ten of a given word.

ALIASES:

DATA ELEMENT NAME: Bit_1

DESCRIPTION: As described in appendix A, specific bits of a word in the emitter track file affect different parameters. Specific bits must be extracted from the track file word before they can be examined. Bit_1 is used to store bit one of a given word.

ALIASES:

DATA ELEMENT NAME: Bit_5

DESCRIPTION: As described in appendix A, specific bits of a word in the emitter track file affect different parameters. Specific bits must be extracted from the track file word before they can be examined. Bit_5 is used to store bit five of a given word.

ALIASES:

DATA ELEMENT NAME: Bit_9

DESCRIPTION: As described in appendix A, specific bits of a word in the emitter track file affect different parameters. Specific bits must be extracted from the track file word before they can be examined. Bit_9 is used to store bit nine of a given word.

ALIASES:

DATA ELEMENT NAME: Blink_circle

DESCRIPTION: The ALR-46 emphasizes certain symbols on the pilot's by placing a blinking circle around the threat's symbol. CGS uses Blink_circle to indicate the threat's symbol should have a blinking circle placed around it.

ALIASES:

DATA ELEMENT NAME: Completed

DESCRIPTION: Used by CGS to indicate the ALR-46 has completed the address changes associated with the track file. Each time the ALR-46 adds a threat, deletes a threat, or changes a threat's priority, and address change occurs.

ALIASES:

DATA ELEMENT NAME: Crot

DESCRIPTION: The CGS name used for the crotale threat.

ALIASES:

DATA ELEMENT NAME: Current_color

DESCRIPTION: Contains the number of the color that will be used to display the next threat symbol. The value of current_color has a range from zero to seven.

ALIASES:

DATA ELEMENT NAME: Current_emitter

DESCRIPTION: Used to point to the Track_file_entry in CGS that it is currently examining.

ALIASES:

Note: Refer to "track_file_entry" for the composition of the records pointed to by Current_emitter.

DATA FLOW NAME: Current_threat

DESCRIPTION: Used to point to the RWR_display_data in CGS that it uses to maintain the information it needs to drive the RWR display.

ALIASES:

COMPOSITION: Current_threat = azimuth + color + new_x_position
new_y_position + old_x_position + old_y_position +
range + status + symbol

DATA ELEMENT NAME: Data

DESCRIPTION: Contains one sixteen bit word from the hardware_monitor_data file.

ALIASES:

DATA ELEMENT NAME: Degrees

DESCRIPTION: Used to store the intermediate values of degrees during the calculation of azimuth.

ALIASES:

DATA ELEMENT NAME: Degree_to_radian

DESCRIPTION: A constant that contains the value to convert degrees to radians.

ALIASES:

DATA ELEMENT NAME: Diamond

DESCRIPTION: Used by CGS to indicate a diamond should be placed around the emitter when it is displayed.

ALIASES:

DATA ELEMENT NAME: Dot

DESCRIPTION: Used by CGS to indicate a dot should be placed inside the emitter symbol. The number of dots which band the threat is transmitting in.

ALIASES:

DATA ELEMENT NAME: Emitter_count

DESCRIPTION: Used by CGS during the initialization of the track_ file_entries to store a different emitter number in each record.

ALIASES:

DATA ELEMENT NAME: Emitter_address

DESCRIPTION: Contains the address of the hardware_monitor word being examined.

ALIASES:

DATA ELEMENT NAME: Emitter_number

DESCRIPTION: Contains the number of the emitter currently being processed by CGS.

ALIASES:

DATA FLOW NAME: Emitter_positions

DESCRIPTION: Contains all fo the information needed by CGS to maintain and display up to sixteen threat symbols on the RWR display.

ALIASES:

COMPOSITION: Emitter_positions = current_threat + current_threat
+ current_threat + current_threat + current_threat
+ current_threat + current_threat + current_threat
+ current_threat + current_threat + current_threat
+ current_threat + current_threat + current_threat
+ current_threat + current_threat

DATA ELEMENT NAME: Emitter_priority

DESCRIPTION: Contains a list of the active emitter sorted by priority starting with the highest priority threat. Each time a new threat is recognized or an existing threat is deactivated, the emitter_priority is re-sorted.

ALIASES:

DATA ELEMENT NAME: Ending_address_of_track_file

DESCRIPTION: Contains the address of the last word in the ALR-46 OFF's track file.

ALIASES:

DATA ELEMENT NAME: Entry_number

DESCRIPTION: Used by CGS in the RWR display to indentify the specific entry on the display being updated.

ALIASES:

DATA ELEMENT NAME: ESC

DESCRIPTION: An ascii control character used by the Tektronix 4027 CRT. It is equal to twenty seven.

ALIASES:

DATA ELEMENT NAME: Hawk

DESCRIPTION: Used by CGS to identify a specific type of threat.

ALIASES:

DATA ELEMENT NAME: Hardware_monitor_data

DESCRIPTION: The name used by CGS to identify the file that
contains the hardware monitor data.

ALIASES:

DATA ELEMENT NAME: Index

DESCRIPTION: Used by CGS as an index into the "emitter_priority".

ALIASES:

DATA ELEMENT NAME: Input

DESCRIPTION: The name of the default input device for CGS.

ALIASES:

DATA ELEMENT NAME: In_progress

DESCRIPTION: Used by CGS to indicate the ALR-46 is in the process
of adding a new threat or deleting an existing one
from its list of active emitters.

ALIASES:

DATA ELEMENT NAME: Jit

DESCRIPTION: Used by CGS to indicate that the pulse train of a
specific threat is jittered.

ALIASES:

DATA ELEMENT NAME: Last_emitter

DESCRIPTION: Is set true to indicate that all of the emitters
have been processed.

ALIASES:

DATA FLOW NAME: List_of_legal_commands

DESCRIPTION: Contains a list of the legal CGS commands.

ALIASES:

COMPOSITION: List_of_legal_commands = Help + Critical + RWR
+ Terminate

DATA ELEMENT NAME: Maximum_value_of_a_byte

DESCRIPTION: The maximum value for a byte is contained in
maximum_value_of_a_byte. It is equal to octal one
hundred and seventy seven.

ALIASES:

DATA ELEMENT NAME: Maximum_number_of_emitters

DESCRIPTION: Is a constant, which contains the maximum number of
emitters the ALR-46 can process. It is initialized
to sixteen.

ALIASES:

DATA ELEMENT NAME: Maximum_RWR_range

DESCRIPTION: Contains the maximum range that can be displayed in the RWR mode.

ALIASES:

DATA ELEMENT NAME: Modified_emitters

DESCRIPTION: Contains a list of the threats that have been modified by CGS during the last time sequence. The list can range from zero to sixteen.

ALIASES:

DATA ELEMENT NAME: Monitor_data

DESCRIPTION: A CGS buffer used to receive a block of data from the hardware monitor file.

ALIASES:

DATA ELEMENT NAME: Monitor_data_index

DESCRIPTION: Used as an index into the monitor_data

ALIASES:

DATA ELEMENT NAME: Name

DESCRIPTION: Used by CGS during the Parameter mode to display the name of the critical parameter being displayed.

ALIASES:

DATA ELEMENT NAME: None_in_progress

DESCRIPTION: Used by CGS to indicate the ALR-46 is not in the process of adding a new threat or deleting an existing threat from it's list of active emitters.

ALIASES:

DATA ELEMENT NAME: Non_active_emitters

DESCRIPTION: Contains a list of the emitters that become in active during the last ALR-46 time sequence. The list can range from zero to sixteen.

ALIASES:

DATA ELEMENT NAME: No_highlighter

DESCRIPTION: Used by CGS to indicate whether or not the emitter being displayed has a highlighter

ALIASES:

DATA ELEMENT NAME: Old_emitters

DESCRIPTION: Contains a list of threats that were active during the last sequence.

ALIASES:

DATA ELEMENT NAME: Out_of_range

DESCRIPTION: Used by CGS to indicate a threat is outside the legal display range.

ALIASES:

DATA ELEMENT NAME: PC_flag

DESCRIPTION: Used to identify the type of hardware monitor data being examined. It is equal to octal 177777. Refer to table 4 for data format.

ALIASES:

DATA ELEMENT NAME: PI

DESCRIPTION: A constant equal to 3.14159.

ALIASES:

DATA ELEMENT NAME: Randians_to_degrees

DESCRIPTION: A CGS constant used to convert radians to degrees.

ALIASES:

DATA ELEMENT NAME: SA_1

DESCRIPTION: The symbol used by CGS to identify a type one surface to air missile (SAM) site.

ALIASES:

DATA ELEMENT NAME: SA_2

DESCRIPTION: The symbol used by CGS to identify a type two surface to air missile site.

ALIASES:

DATA ELEMENT NAME: SA_3

DESCRIPTION: The symbol used by CGS to identify a type three surface to air missile site.

ALIASES:

DATA ELEMENT NAME: SA_4

DESCRIPTION: The symbol used by CGS to identify a type four surface to air missile site.

ALIASES:

DATA ELEMENT NAME: SA_5

DESCRIPTION: The symbol used by CGS to identify a type five surface to air missile site.

ALIASES:

DATA ELEMENT NAME: SA_6

DESCRIPTION: The symbol used by CGS to identify a type six surface to air missile site.

ALIASES:

DATA ELEMENT NAME: SA_7

DESCRIPTION: The symbol used by CGS to identify a type seven surface to air missile site.

ALIASES:

DATA ELEMENT NAME: SA_8

DESCRIPTION: The symbol used by CGS to identify a type eight surface to air missile site.

ALIASES:

DATA ELEMENT NAME: SA_9

DESCRIPTION: The symbol used by CGS to identify a type nine surface to air missile site.

ALIASES:

DATA ELEMENT NAME: Stab

DESCRIPTION: Used by CGS to indicate the threat's pulse train is stable.

ALIASES:

DATA ELEMENT NAME: Staf2

DESCRIPTION: Used by CGS to identify a special type two surface to air missile site.

ALIASES:

DATA ELEMENT NAME: Staf6

DESCRIPTION: Used by CGS to identify a special type six surface to air missile site.

ALIASES:

DATA ELEMENT NAME: Staf8

DESCRIPTION: Used by CGS to identify a special type eight surface
to air missile site.

ALIASES:

DATA ELEMENT NAME: Stag

DESCRIPTION: Used by CGS to indicate the threat's pulse train is
staggered.

ALIASES:

DATA ELEMENT NAME: Steady_circle

DESCRIPTION: Used by CGS to indicate the threat symbol should
have a steady circle placed around it as a high-
lighter.

ALIASES:

DATA FLOW NAME: Track_file_free_pointer

DESCRIPTION: Contains the address used by the ALR-46 to point to
to the first free record in its emitter track file.

ALIASES:

DATA ELEMENT NAME: Track_file_used_pointer

DESCRIPTION: Contains the address of the word used by the ALR-46
to point to the first used record in its emitter
track file.

ALIASES:

DATA FLOW NAME: Track_file_pointer

DESCRIPTION: Points to specific entry in the CGS track file.

ALIASES:

COMPOSITION: Track_file_pointer = azimuth + cd_age_counter +
cepc + deviation + emitter_type + high_lighter +
hi_band_age_cnt + hi_band_cos + hi_band_sin +
lethality_ring + ml_age + next_emitter + power_
pround + priority + prio_poweround + pri_average
+ pri_frame_period + pules_train_descriptor +
range + record_number + rf_band + special_field +
stagger_level + symbol + updated_entries +
x_displacement + y_displacement

DATA ELEMENT NAME: Starting_address_of_track_file

DESCRIPTION: Contains the address of the first word in the
ALR-46 OFF's track file.

ALIASES:

DATA ELEMENT NAME: Unk0

DESCRIPTION: Used in CGS as a label for a threat that the ALR-46
is unable to identify. The "Unknown" threat is
transmitting in band zero.

ALIASES:

DATA ELEMENT NAME: Unk1

DESCRIPTION: Used in CGS as a label for a threat that the ALR-46
is unable to identify. The "Unknown" threat is
transmitting in band one.

ALIASES:

DATA ELEMENT NAME: Unk2

DESCRIPTION: Used in CGS as a label for a threat that the ALR-46 is unable to identify. The "Unknown" threat is transmitting in band two.

ALIASES:

DATA ELEMENT NAME: Unk3

DESCRIPTION: Used in CGS as a label for a threat that the ALR-46 is unable to identify. The "Unknown" threat is transmitting in band three.

ALIASES:

DATA ELEMENT NAME: Unk4

DESCRIPTION: Used in CGS as a label for a threat that the ALR-46 is unable to identify. The "Unknown" threat is transmitting in band four.

ALIASES:

DATA ELEMENT NAME: Vertical_line_count

DESCRIPTION: Used by CGS to count the number of vertical lines during the initialization of the VT-100 CRT.

ALIASES:

DATA ELEMENT NAME: Wing

DESCRIPTION: Used in CGS to identify a specific airborne threat

ALIASES:

DATA ELEMENT NAME: Word_number

DESCRIPTION: Contains the number of the word being examined from a CGS track file entry. It has a range from zero to fifteen.

ALIASES:

DATA ELEMENT NAME: X_scale_factor

DESCRIPTION: Is a percent of the maximum distance, from the origin of the RWR display, a threat can be placed. For example, if x_scale_factor was equal to one hundred percent, then the symbol would be placed at the intersection of the x axis and the outer most ring.

ALIASES:

DATA ELEMENT NAME: Y_scale_factor

DESCRIPTION: Is a percent of the maximum distance, from the origin of the RWR display, a threat can be placed. For example, if y_scale_factor was equal to one hundred percent, then the symbol would be placed at the intersection of the y axis and the outer most ring.

ALIASES:

APPENDIX G

CGS TEST DOCUMENTATION

The data used to validate CGS was divided into two separate files, one dedicated to the validation of the Parameter mode and the other dedicated to the RWR mode. Two data bases allowed data to be selected that made it easy to visually validate each mode by examining the information displayed on the operator's CRT. For example, the record number data values selected for the Parameter mode resulted in sixteen record numbers ranging from one to sixteen. Both the Parameter mode and the RWR mode data bases were divided into groups, one for each type of word processed by CGS. Each group can be easily identified by the number which precedes it. For example, the data words for word number two are preceded by the number two entered four times. The results of the execution of CGS with the parameter data base is shown in Table 8. The results of the execution of CGS with the RWR data base is shown in Table 9. The ALR-46 test data base follows Table 9.

Table 8. Parameter Display Test Results

BY PRIORITY	1	2	3	16
SYMBOL	SA_1	.. SA_9	HAWK	BAR_2	CROT	WING	AIR	AAA_0	AAA_1							
TYPE	101	102	103	104	105	116
BANDS	1	2	3	4	0	0
PRI / FRAME PER	1.1	2.2	3.3	4.4	..9.9	11.1	77.7
PULSE TRAIN DES	2STAG	3STAG	8STAG	STAB	STAB
PRI AVERAGE	1.1	2.2	3.3	.	.	9.9	1.1	7.7
DEVIATION	1.0	2.0	3.0	16.0
SPECIAL	YY	NN	YY.	NN
POWER/PWR ROUND	1	2	3	16
HIGH BAND SINE	1	2	3	16
HIGH BAND COS	1	2	3	16
CEPC	1	2	3	16
HI BAND AGE CNT	1	2	3	16
CD BAND AGE CNT	1	2	3	16
PRIORITY	1	2	3	16
PRIORITY /ROUND	1	2	3	16
AZIMUTH	0	22	45.	337
LETHALITY RING	0	0	0	0
ML AGE	1	2	3	16
RECORD NUMBER	1	2	3	16

Table 9. RWR Display Test Results

SYMBOL	RANGE	AZIMUTH
SA_1	100	0
SA_2	100	22
:	:	:
:	:	:
SA_9	100	180
HAWK	100	202
BAR_2	100	225
CROT	100	247
WING	100	270
AIR	100	292
AAA_0	100	315
AAA_1	100	337
SA_1	100	0
SA_2	100	22
SA_3	100	45
WING	100	67
AIR	100	90
AAA_0	100	112
AAA_1	100	135
AAA_2	100	157
AAA_3	100	180
STAF2	100	202
STAF6	100	225
STAF8	100	247
UNK0	100	270
UNK1	100	292
UNK2	100	315
UNK3	100	337

Parameter Test Data Base

0
0
0
0
5000
45
5016
5000
5032
5016
5048
5032
5064
5048
5080
5064
5096
5080
5112
5096
5128
5112
5144
5128
5160
5144
5176
5160
5192
5176
5208
5192
5224
5208
5240
5224
45
5240
2
2
2
2
8001
5002
8002
5018
8003
5034
8004
5050

Parameter Test Data Base

8005
5066
8006
5082
8007
5098
8008
5114
8009
5130
8010
5146
8011
5162
8012
5178
8013
5194
8014
5210
8015
5226
8016
5242
3
3
3
3
11
5003
22
5019
33
5035
44
5051
55
5067
66
5083
77
5099
88
5115
99
5131
111
5147
222
5163

Parameter Test Data Base

333
5179
444
5195
555
5211
666
5227
777
5243
4
4
4
4
256
5004
512
5020
768
5036
1024
5052
1280
5068
1536
5084
1792
5100
2048
5116
2304
5132
2560
5148
2816
5164
3072
5180
3328
5196
3584
5212
3840
5228
4096
5244
5
5
5
5

Parameter Test Data Base

256
5005
512
5021
768
5037
1024
5053
1280
5069
1536
5085
1792
5101
2048
5117
2304
5133
2560
5149
2816
5165
3072
5181
3328
5197
3584
5213
3840
5229
4096
5245
6
6
6
6
1057
5006
2114
5022
3171
5038
4228
5054
5288
5070
6336
5086
7392
5102

Parameter Test Data Base

8448
5118
9504
5134
10560
5150
11616
5166
12672
5182
13728
5198
14784
5214
15840
5230
16896
5246
7
7
7
7
1025
5007
2050
5023
3076
5039
4104
5055
5120
5071
6144
5087
7168
5103
8192
5119
9216
5135
10240
5151
11264
5167
12288
5183
13312
5199
14336
5215

Parameter Test Data Base

15872
5231
16384
5247
8
8
8
8
2337
5008
4674
5024
7011
5040
9348
5056
11685
5072
14022
5088
16359
5104
16384
5120
18432
5136
20480
5152
22528
5168
24576
5184
26624
5200
28672
5216
30720
5232
32768
5248
9
9
9
9
4480
5009
5888
5025
6400
5041

Parameter Test Data Base

5888
5057
4480
5073
2432
5089
0
5105
5728
5121
3680
5137
2272
5153
1760
5169
2272
5185
3680
5201
5728
5217
0
5233
2432
5249
10
10
10
10
4480
5010
2432
5026
0
5042
5728
5058
3680
5074
2272
5090
1760
5106
2272
5122
3680
5138
5728
5154

Parameter Test Data Base

0
5170
2432
5186
4480
5202
5888
5218
6400
5234
5888
5250
12
12
12
12
11
5012
22
5028
33
5044
44
5060
55
5076
66
5092
77
5108
88
5124
99
5140
11
5156
22
5172
33
5188
44
5204
55
5220
66
5236
77
5252
13
13

Parameter Test Data Base

13
13
65
5013
128
5029
193
5045
256
5061
321
5077
384
5093
449
5109
512
5125
577
5141
641
5157
705
5173
768
5189
833
5205
896
5221
961
5237
1024
5253
14
14
14
14
8
5014
16
5030
24
5046
32
5062
40
5078
48
5094

Parameter Test Data Base

56
5110
64
5126
72
5142
80
5158
88
5174
96
5190
104
5206
112
5222
120
5238
128
5254
15
15
15
15
1
5015
2
5031
3
5047
4
5063
5
5079
6
5095
7
5111
8
5127
9
5143
10
5159
11
5175
12
5191
13
5207

Parameter Test Data Base

14
5223
15
5239
16
5255
45
5192
15
5239
16
5255

RWR Test Data Base

5000
45
5016
5000
5032
5016
5048
5032
5064
5048
5080
5064
5096
5080
5112
5096
5128
5112
5144
5128
5160
5144
5176
5160
5192
5176
5208
5192
5224
5208
5240
5224
45
5240
9
9
9
9
10
10
10
10
8001
5002
4480
5009
4480
5010
8002
5018

RWR Test Data Base

5888
5025
2432
5026
8003
5034
6400
5041
0
5042
8004
5050
5888
5057
5728
5058
8005
5066
4480
5073
3680
5074
8006
5082
2432
5089
2272
5090
8007
5098
0
5105
1760
5106
8008
5114
5728
5121
2272
5122
8009
5130
3680
5137
3680
5138
8010
5162
2272
5153

RWR Test Data Base

5728
5154
8011
5162
1760
5169
0
5170
8012
5178
2272
5185
2432
5186
8013
5194
3680
5201
4480
5202
8014
5210
5728
5217
5888
5218
8015
5226
0
5233
6400
5234
8016
5242
2432
5249
5888
5250
0
0
0
0
45
5224
0
0
45
5208
0
0

RWR Test Data Base

45
5192
0
0
45
5176
0
0
45
5160
0
0
45
5144
0
0
45
5128
0
0
45
5112
0
0
45
5096
0
0
45
5080
0
0
45
5064
0
0
45
5048
0
0
45
5032
0
0
45
5016
0
0
45
5000

RWR Test Data Base

0
0
45
45
0
0
0
0
9
9
9
9
10
10
10
10
8001
5002
4480
5009
4480
5010
8002
5018
5888
5025
2432
5026
8003
5034
6400
5041
0
5042
8013
5050
5888
5057
5728
5058
8014
5066
4480
5073
3680
5074
8015
5082
2432
5089

RWR Test Data Base

2272
5090
8016
5098
0
3105
1760
5106
8017
5114
5728
5121
2272
5122
8018
5130
3680
5137
3680
5138
8019
5162
2272
5153
5728
5154
8020
5162
1760
5169
0
5170
8021
5178
2272
5185
2432
5186
8022
5194
3680
5201
4480
5202
8023
5210
5728
5217
5888
5218

RWR Test Data Base

8024
5226
0
5233
6400
5234
8025
5242
2432
5249
5888
5250
0
0
0
0

RWR Test Data Base

Vita

Mr. J. Wayne Thames was born on March 9, 1949, in Fort Valley, Georgia. In 1967, he graduated from Fort Valley High School in Fort Valley, Georgia. He attended Auburn University from which he received a Bachelor of Science in Electrical Engineering degree in 1971. Following graduation, he accepted a position with the General Electric Company, Electrolytic Capacitor Division. He was employed by G.E. as an electronic design engineer until 1973 when he accepted employment with the Electronic Warfare Division of the United States Air Force as a civilian electronic engineer working on the development of the Electronic Warfare Open Loop Simulator. He entered the Air Force Institute of Technology at Wright Patterson AFB, Ohio in June 1980.

Permanent address: Route 3 Box 57-F

Fort Valley, GA. 31030

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
AFIT/GCS/EE/81D-18		
4. TITLE (and Subtitle)		5. TYPE OF REPORT & PERIOD COVERED
ALR-46 COMPUTER GRAPHICS SYSTEM FOR THE ROBINS AFB ELECTRONIC WARFARE DIVISION ENGINEERING BRANCH LABORATORY		MS THESIS
7. AUTHOR(s)		6. PERFORMING ORG. REPORT NUMBER
J. Wayne Thames, Civ, USAF		
9. PERFORMING ORGANIZATION NAME AND ADDRESS		8. CONTRACT OR GRANT NUMBER(s)
Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH 45433		
11. CONTROLLING OFFICE NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH 45433		
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE
		December 1981
		13. NUMBER OF PAGES
		231
		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
<p style="text-align: right;">LYNN E WOLAVER</p> <p style="text-align: center;">8 SEP 1982</p> <p>IAW AFR (90-17)</p> <p>Approved for public release; Frederic C. Lynch, Major, USAF Dean for Research and Professional Development Director of Public Affairs Air Force Institute of Technology (ATC) Wright-Patterson AFB, OH 45433</p>		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
Computer Graphics System ALR-46		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
See Reverse		

DD FORM 1473
1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Continuation of Block 20

An ALR-46 Computer Graphics System (CGS) for the Electronic Warfare Division Engineering Branch Laboratory was designed and implemented. The system is composed of the ALR-46 Radar Warning Receiver; real time hardware monitor; PDP 11/34 computer; and two display devices, a Tektronix model 4027 CRT and a Digital Equipment Corporation (DEC) model VT-100 CRT. The ALR-46 test engineers at the Engineering Branch laboratory were interviewed; their functional requirements were translated into a detailed set of hardware and software requirements. Structured Analysis Techniques were used to produce a structured specification for the system requirements. Yourdon and Constantine's Transform and Transaction Analysis techniques were used to develop module structure charts for the software design. The final software design phase translated the module structure charts into Warnier-Orr (W/O) diagrams. These were translated into operational software using DEC Pascal. The software was implemented and tested using a top down approach. The modules and data structures were designed to allow additional capabilities to be added to CGS with minimum impact on the current system. The data acquired by the hardware monitor from the ALR-46 is passed to the PDP 11/34 for CGS analysis. This information is used either to simulate the pilot's display or to generate a critical parameter display describing the threats being processed by the ALR-46.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

LMEL
-8